810project

Team2: Chiebuka Onwuzurike, Tzu-Hua Huang, Yangyang Zhou, Yichi Zhan

03/02/2021

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 (http://rmarkdown.rstudio.com).

Install packages

```
# install.packages("corrplot")
# install.packages("googleway")
# install.packages("caret")
# install.packages("lattice")
# install.packages("standardize")
# install.packages('mice')
# install.packages("gridExtra")
# install.packages("rpart")
# install.packages("rpart.plot")
# install.packages("Metrics")
# install.packages("dplyr")
# install.packages("ipred")
# install.packages(c("gbm"))
# install.packages("randomForest")
```

```
# if(!requireNamespace("devtools")) install.packages("devtools")
# devtools::install_github("dkahle/ggmap", ref = "tidyup", force=TRUE)
```

Setup

```
library(data.table)
library(ggplot2)
library(ggthemes)
library(glmnet)
library(corrplot)
library(lattice)
library(caret)
library(standardize)
library(ggmap)
library(scales)
library(gridExtra)
library(rpart)
library(rpart.plot)
library(Metrics)
library(dplyr)
library(ipred)
library(gbm)
library(randomForest)

theme_set(theme_bw())
```

```
# Using API from GCP to get the map
ggmap::register_google(key = "AIzaSyCkXRM7Y48Tu8kUNsm4jVkpscP9dFoESzg")
```

Loading data

```
# dd <- fread("Melbourne_housing_FULL.csv")
dd <-
  fread(
    "/Users/huangtzuhua/Documents/2021_BU MSBA/2101_BA 810/Team Project/Dataset/Melbo
urne_housing_FULL.csv"
  )

# data size and structure
dim(dd)
```

```
## [1] 34857    21
```

```
str(dd)
```

```
## Classes 'data.table' and 'data.frame':   34857 obs. of  21 variables:
##  $ Suburb       : chr  "Abbotsford" "Abbotsford" "Abbotsford" "Abbotsford" ...
##  $ Address      : chr  "68 Studley St" "85 Turner St" "25 Bloomburg St" "18/659 Vi
ctoria St" ...
##  $ Rooms        : int  2 2 2 3 3 3 4 4 2 2 ...
##  $ Type         : chr  "h" "h" "h" "u" ...
##  $ Price        : int  NA 1480000 1035000 NA 1465000 850000 1600000 NA NA NA ...
##  $ Method       : chr  "SS" "S" "S" "VB" ...
##  $ SellerG      : chr  "Jellis" "Biggin" "Biggin" "Rounds" ...
##  $ Date         : chr  "3/09/2016" "3/12/2016" "4/02/2016" "4/02/2016" ...
##  $ Distance     : num  2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 ...
##  $ Postcode     : num  3067 3067 3067 3067 3067 ...
##  $ Bedroom2     : int  2 2 2 3 3 3 3 3 4 3 ...
##  $ Bathroom     : int  1 1 1 2 2 2 1 2 1 2 ...
##  $ Car          : int  1 1 0 1 0 1 2 2 2 1 ...
##  $ Landsize     : int  126 202 156 0 134 94 120 400 201 202 ...
##  $ BuildingArea : num  NA NA 79 NA 150 NA 142 220 NA NA ...
##  $ YearBuilt    : int  NA NA 1900 NA 1900 NA 2014 2006 1900 1900 ...
##  $ CouncilArea  : chr  "Yarra City Council" "Yarra City Council" "Yarra City Counc
il" "Yarra City Council" ...
##  $ Lattitude    : num  -37.8 -37.8 -37.8 -37.8 -37.8 ...
##  $ Longtitude   : num  145 145 145 145 145 ...
##  $ Regionname   : chr  "Northern Metropolitan" "Northern Metropolitan" "Northern M
etropolitan" "Northern Metropolitan" ...
##  $ Propertycount: num  4019 4019 4019 4019 4019 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Data Cleaning

- We choose to remove all observation without Price, Landsize and Yearbuilt, because we think those are important criterias to make predictions.
- There are more than 60% of data missing in the BuildingArea columns. Since building area and landsize are highly correlated, we choose to drop the BuildingArea column.
- We fill the null values in the Car column with 0, because those houses or apartments do not have parking lots.

```
# view missing values
sum(is.na(dd))
```

```
## [1] 100969
```

```
colSums(is.na(dd))
```

```
##        Suburb       Address         Rooms          Type         Price
##             0             0             0             0          7610
##        Method       SellerG          Date      Distance      Postcode
##             0             0             0             1             1
##      Bedroom2      Bathroom           Car      Landsize  BuildingArea
##          8217          8226          8728         11810         21115
##     YearBuilt   CouncilArea     Lattitude    Longtitude    Regionname
##         19306             0          7976          7976             0
## Propertycount
##             3
```

```
# removing missing values
dd <- dd[!(is.na(dd$Price))]
dd <- dd[,BuildingArea:=NULL]
dd <- dd[!is.na(dd$Landsize)]
dd <- dd[!is.na(dd$YearBuilt)]
dd$Car[which(is.na(dd$Car))] <- 0
dd <- dd[!is.na(dd$Lattitude)]
colSums(is.na(dd))
```

```
##        Suburb       Address         Rooms          Type         Price
##             0             0             0             0             0
##        Method       SellerG          Date      Distance      Postcode
##             0             0             0             0             0
##      Bedroom2      Bathroom           Car      Landsize     YearBuilt
##             0             0             0             0             0
##   CouncilArea     Lattitude    Longtitude    Regionname Propertycount
##             0             0             0             0             0
```

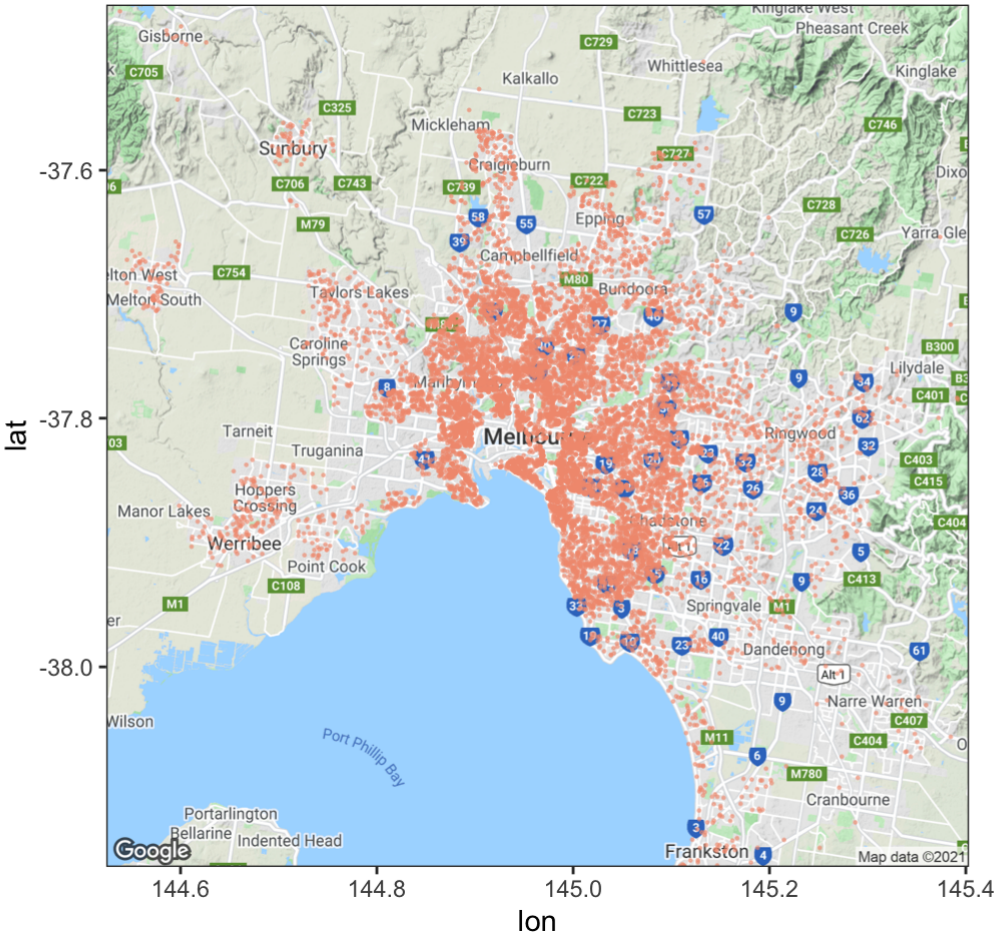
```
dim(dd)
```

```
## [1] 10613    20
```

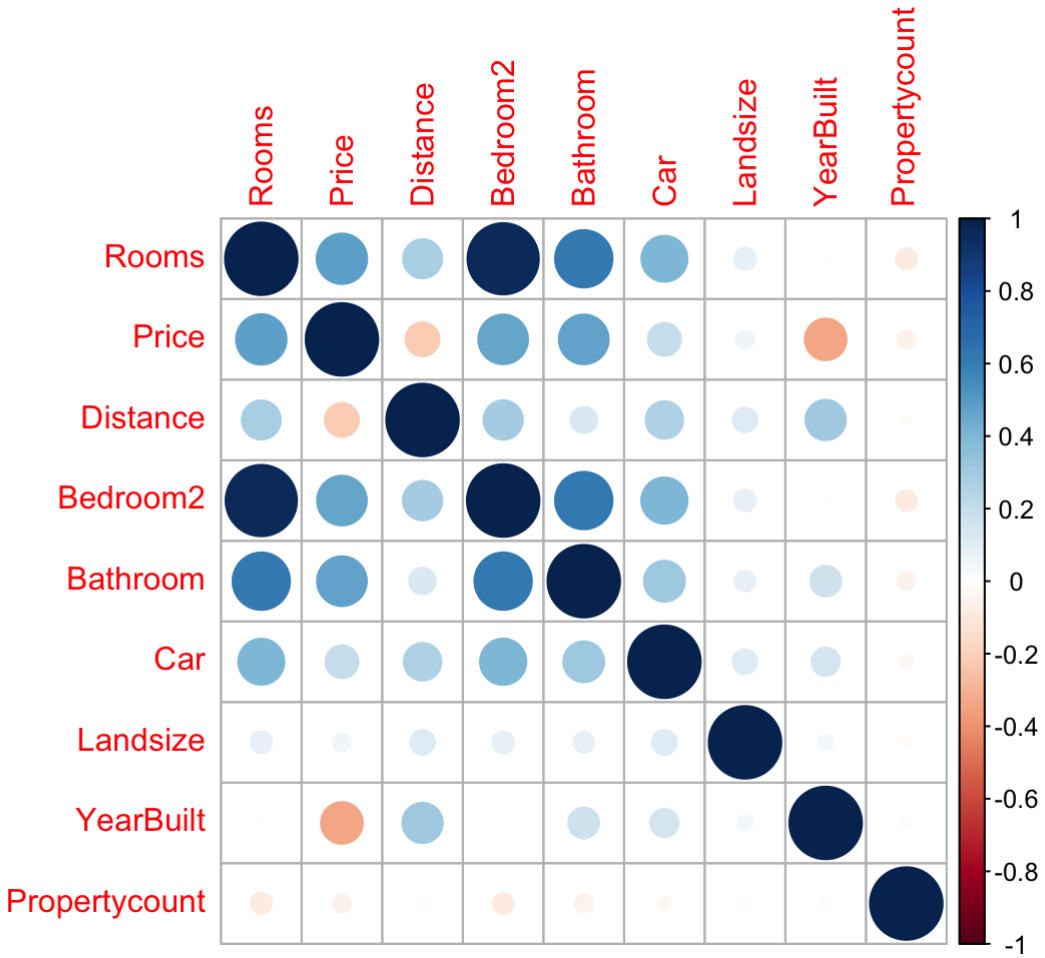
EDA

Using API form GCP to get the map and see where are those houses located.

```
ggmap(get_googlemap(center = c(lon = 144.9631, lat = -37.8136),
      zoom = 10, scale = 2,
      maptype ='terrain',
      color = 'color')) +
  geom_point(aes(x = Longtitude, y = Lattitude),colour='#F39B7FB2', data = dd, size =
0.2, alpha = 0.5) +
  theme(legend.position="bottom")
```



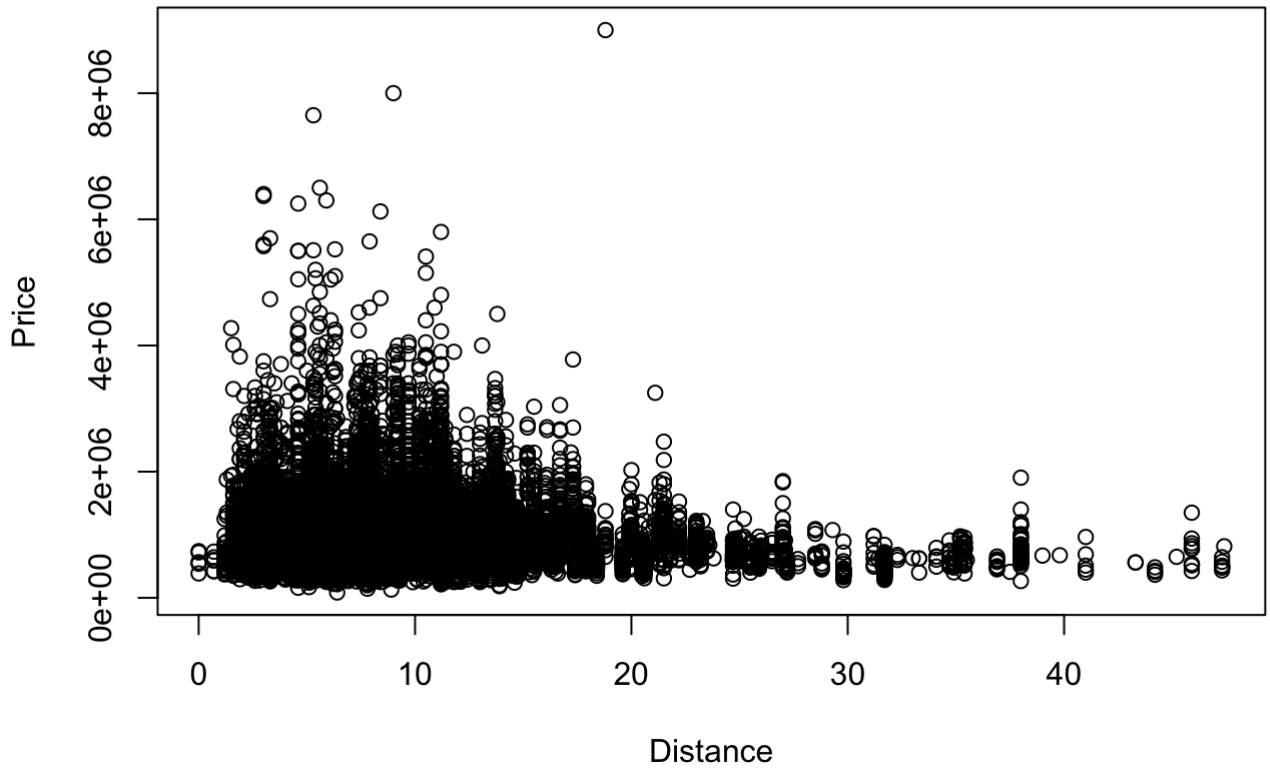

```
# checking correlations between numerical data
mel_n <- dd[,c(3,5,9,11,12,13,14,15,20)]
corrplot(cor(mel_n))
```



Interpretation

- From the plot we notice that the number of bedroom is highly correlated to the number of rooms.
- Rooms and price has strong positive relationship. Houses with more rooms have higher price.
- YearBuilt and price have positive relationship, which indicates as the older building tends to have higher price.
- Houses that are far from CBD tend to have lower price.

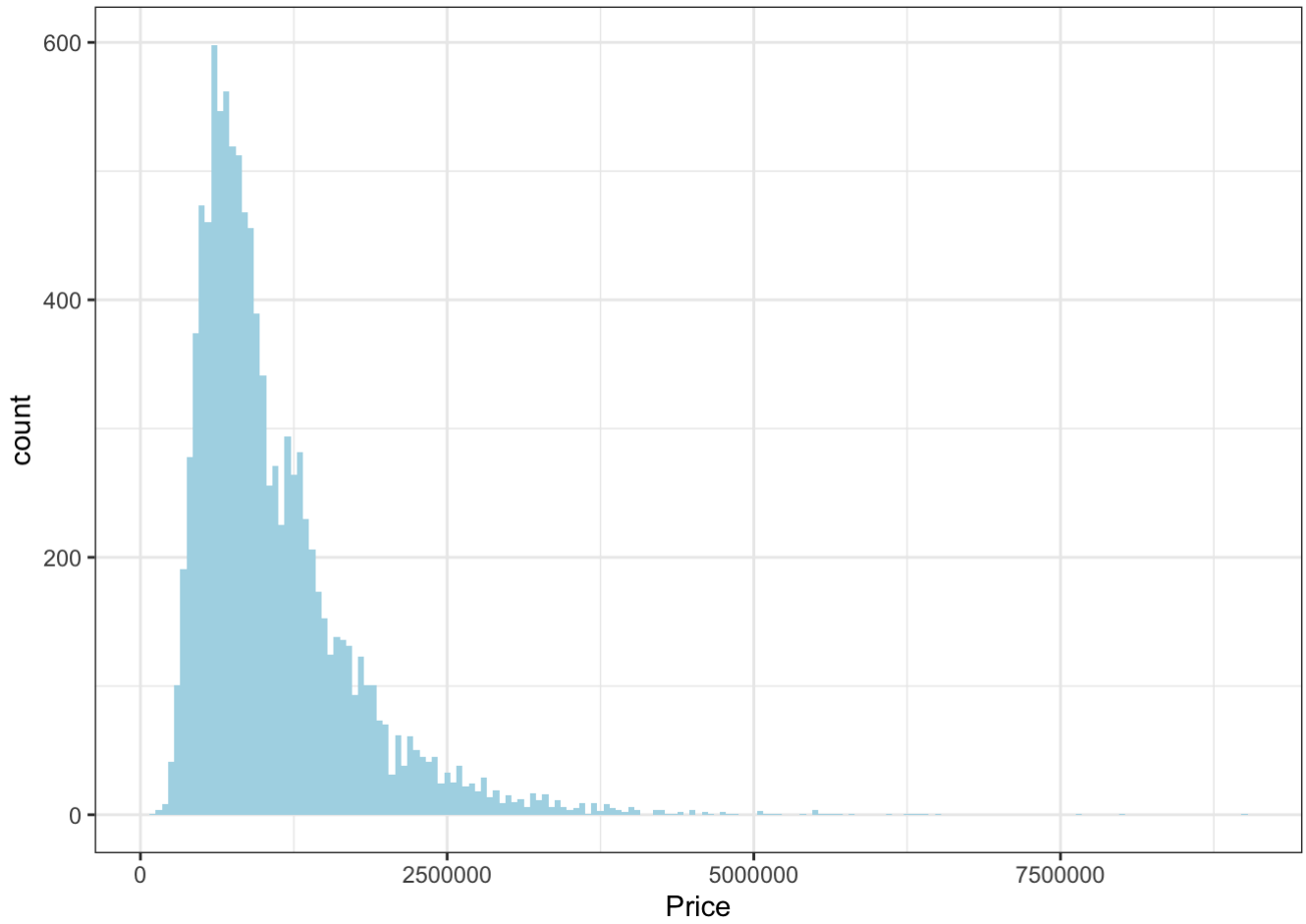
```
plot(Price ~ Distance, data = dd)
```



Interpretation

- Houses that are far from CBD tend to have lower price.

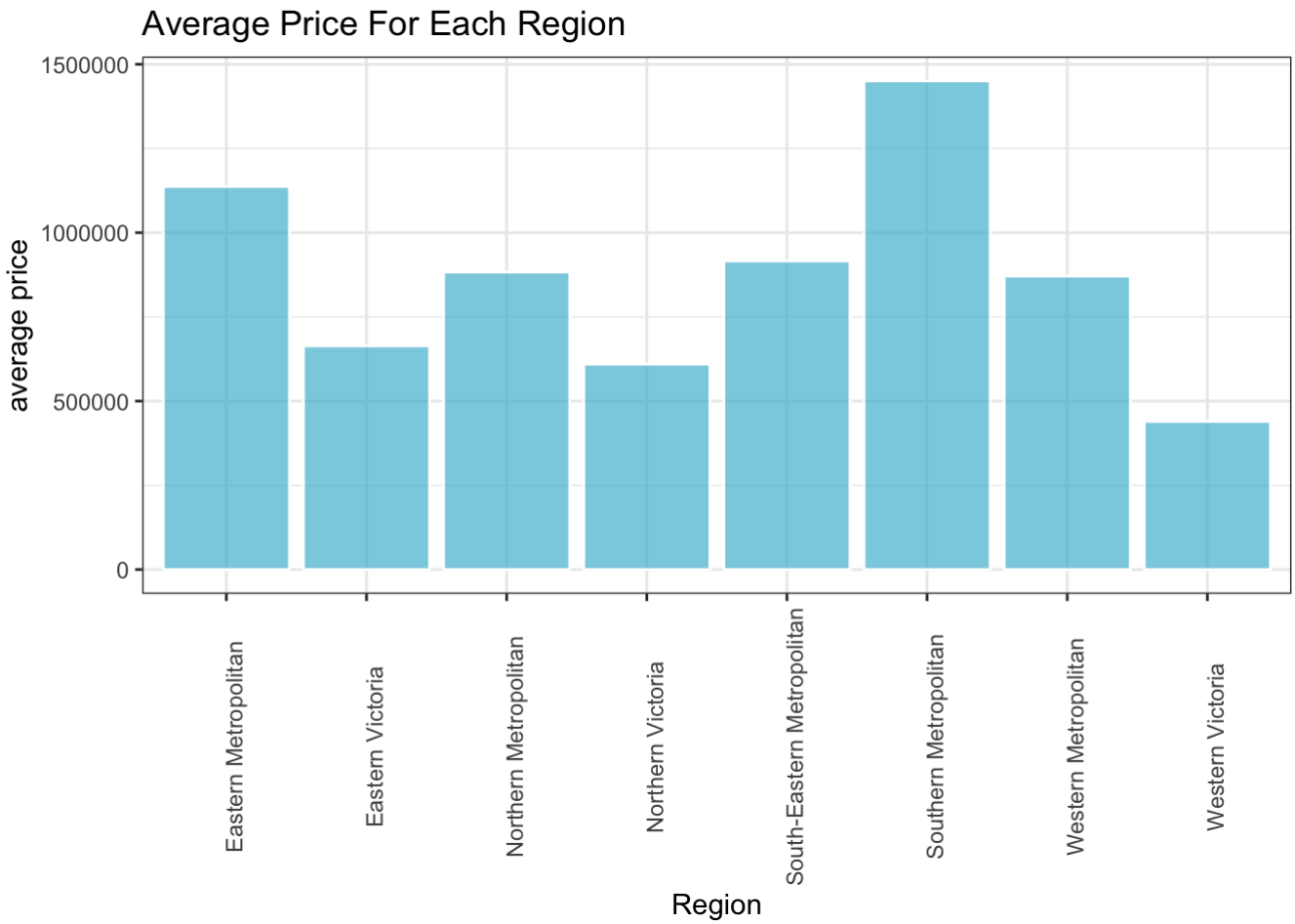
```
# The price distribution
ggplot(data=dd,aes(x=Price))+
  geom_histogram(binwidth=50000,fill='lightblue')
```



Interpretation

- Most of the houses are under $1,250,000
- There are some outliers have really high prices

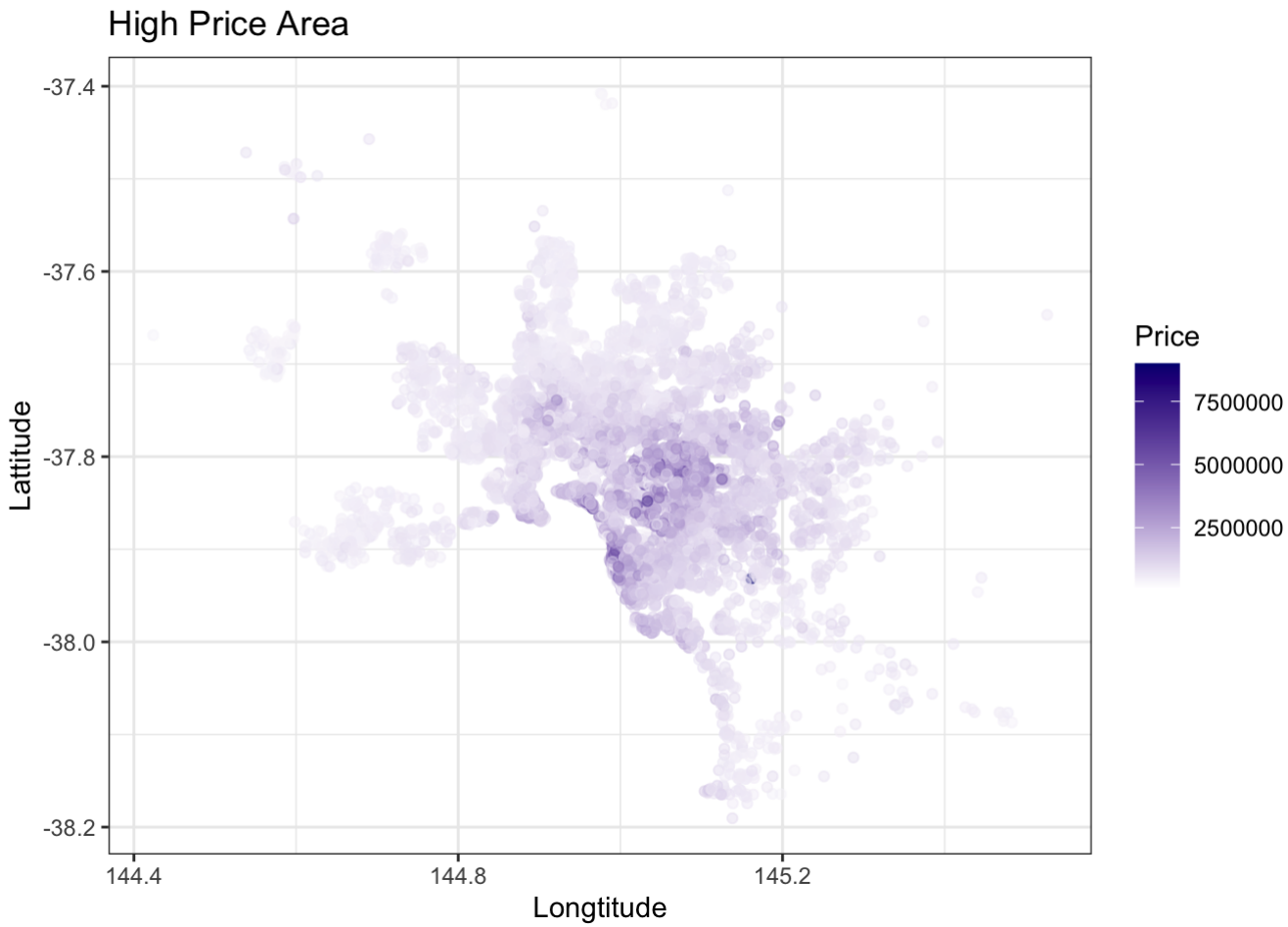
```
# average price for each region
avg_price = dd[ ,mean(Price) ,by=Regionname]
ggplot(avg_price, aes(x=Regionname, y=V1)) +
  geom_bar(stat='identity',colour="white", fill="#4DBBD5B2") +
  labs(x="Region", y="average price", title = "Average Price For Each Region") +
  theme(axis.text.x = element_text(angle = 90))
```



Interpretation

- Southern Metropolitan, has the highest average price overall.
- Western Victoria, has the lowest average price.

```
# Highest price regions
ggplot(data = dd, aes(y=Lattitude, x=Longtitude)) +
  geom_point(aes(colour=Price), alpha=0.5) +
  scale_colour_gradient(low = "white", high = "navy") +
  labs(title="High Price Area")
```



Interpretation

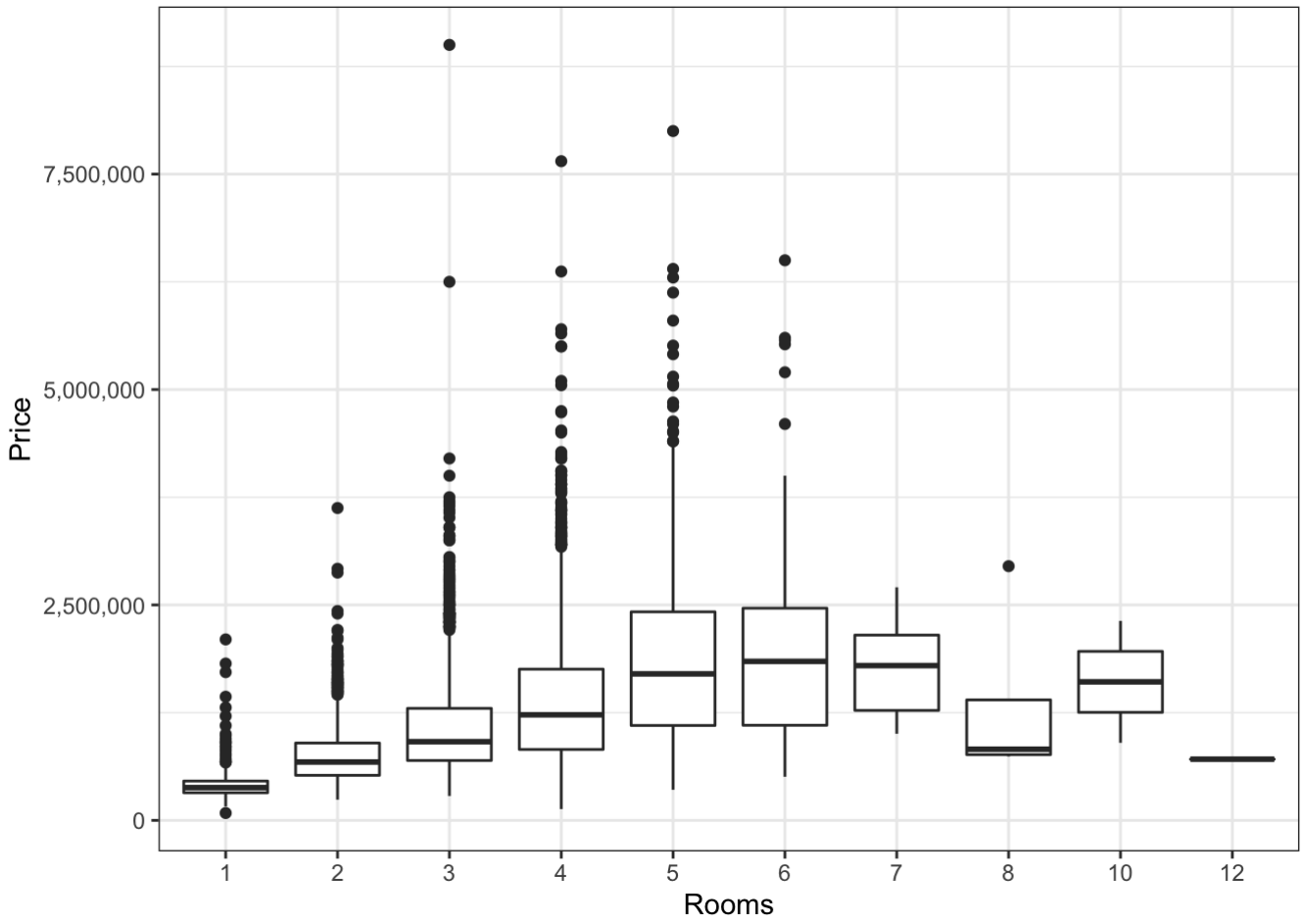
- Houses in the center of Melbourne and along the Port Phillip Bay tend to have higher price.

```
# Price differences for each type of houses
ggplot(dd, aes(x=Propertycount, y=Price)) +
  geom_point(aes(colour=Type), alpha=0.4) +
  theme_bw() + xlab("Property count") +
  labs(y = "Price", title = "Houses have higher price")
```

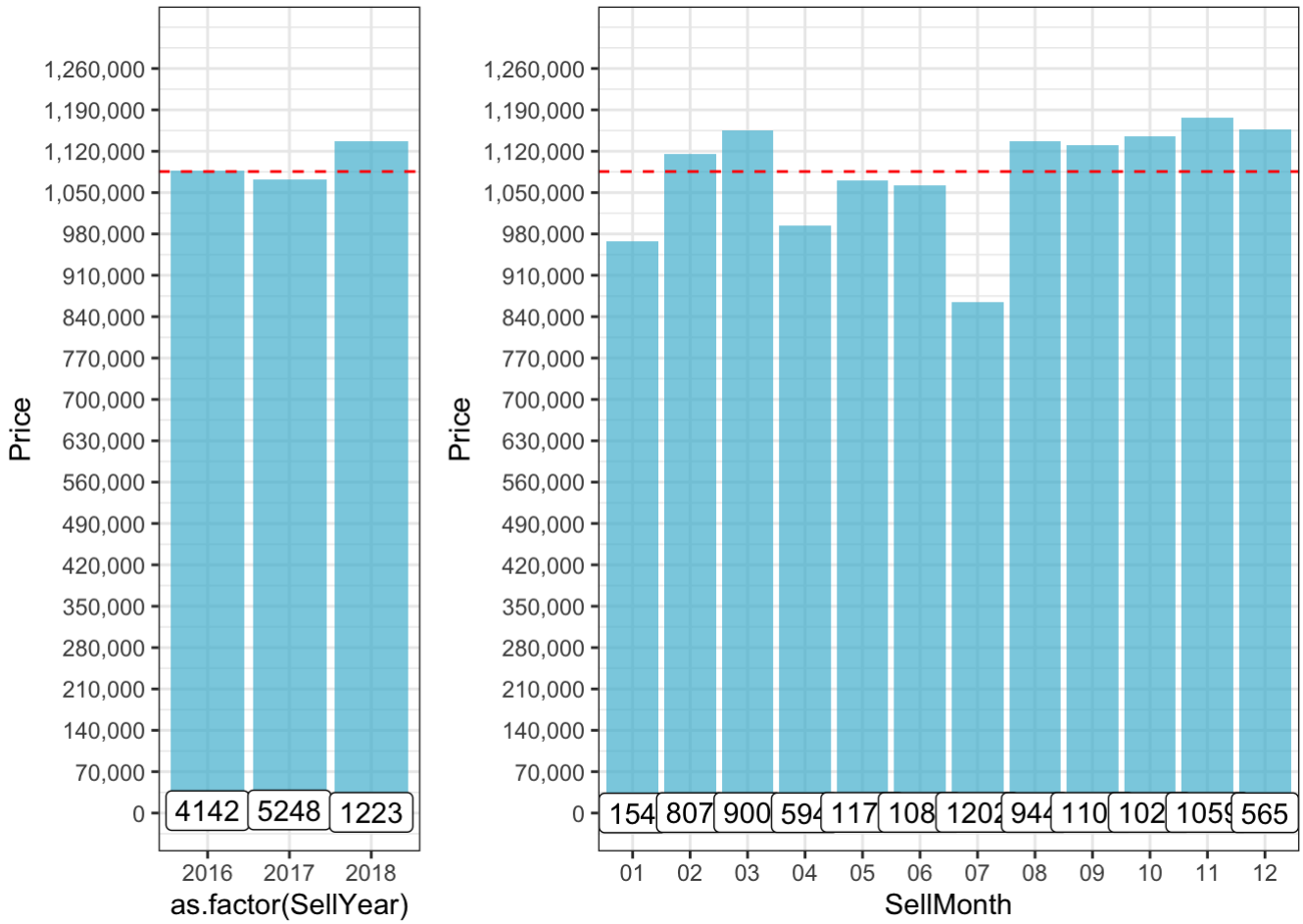
Interpretation

- The majority of our samples are houses
- House has higher price than townhouse and unit

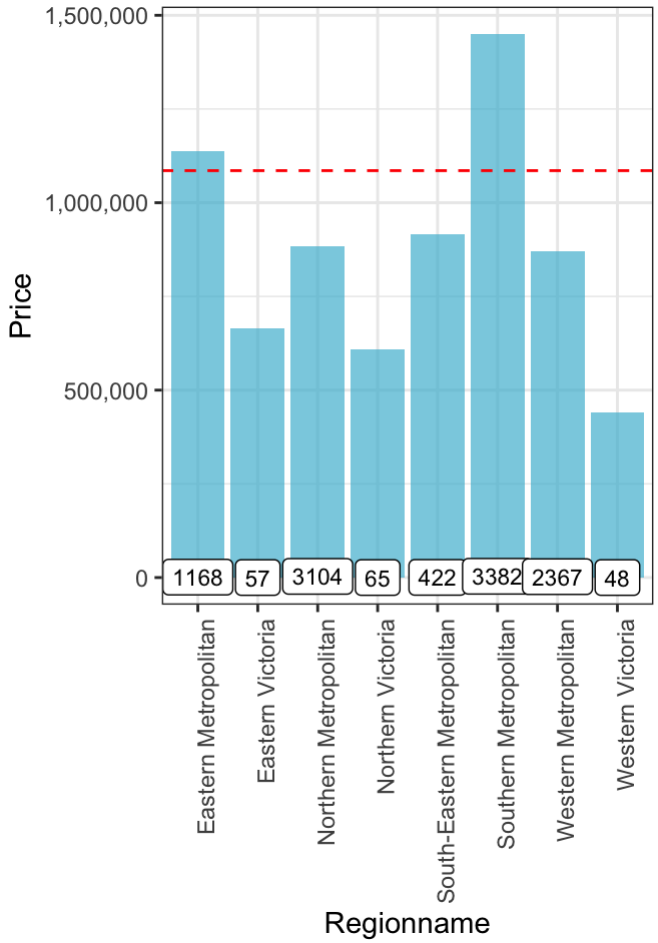
```
ggplot(data=dd, aes(x=factor(Rooms), y=Price))+
        geom_boxplot() + labs(x='Rooms') +
        scale_y_continuous(labels = comma)
```

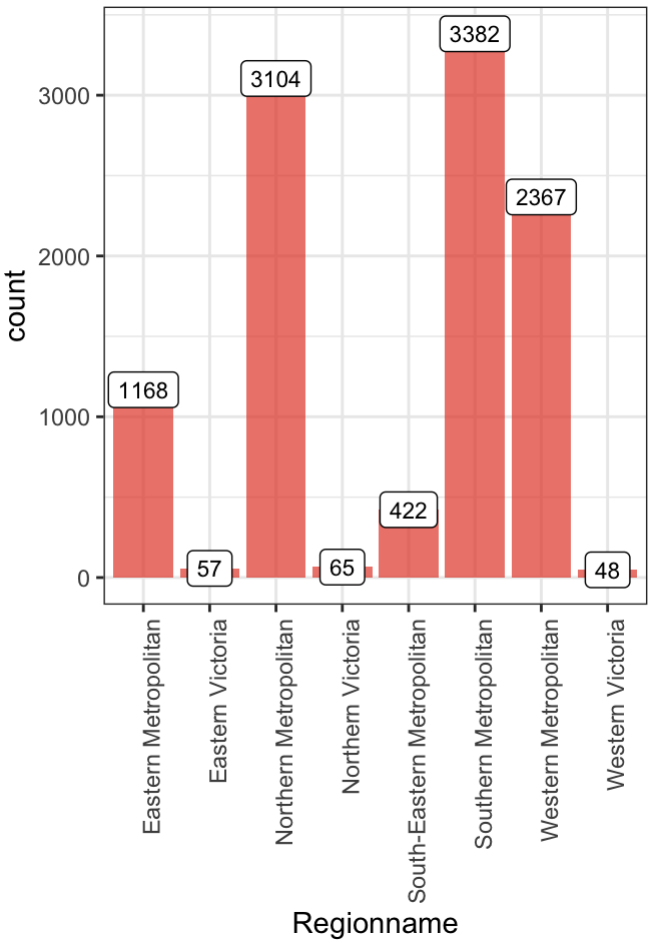



```
dd$Date <- as.Date(dd$Date,"%d/%m/%Y")
dd$SellYear <- format(as.Date(dd$Date),"%Y")
dd$SellMonth <- format(as.Date(dd$Date),"%m")

ys <- ggplot(dd, aes(x=as.factor(SellYear), y=Price)) +
        geom_bar(stat='summary', fun = "mean", fill='#4DBBD5B2')+
        scale_y_continuous(breaks= seq(0, 1300000, by=70000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..)) +
        coord_cartesian(ylim = c(0, 1300000)) +
        geom_hline(yintercept=1085593, linetype="dashed", color = "red")
#dashed line is median Price
ms <- ggplot(dd[!is.na(dd$Price),], aes(x=SellMonth, y=Price)) +
        geom_bar(stat='summary', fun = "mean", fill='#4DBBD5B2')+
        scale_y_continuous(breaks= seq(0, 1300000, by=70000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..)) +
        coord_cartesian(ylim = c(0, 1300000)) +
        geom_hline(yintercept=1085593, linetype="dashed", color = "red")
#dashed line is median Price

grid.arrange(ys, ms, widths=c(1,2))
```




```
n1 <- ggplot(dd, aes(x=Regionname, y=Price)) +
        geom_bar(stat='summary', fun.y = "mean", fill='#4DBBD5B2') +
        theme(axis.text.x = element_text(angle=90, hjust = 1)) +
        scale_y_continuous(breaks= seq(0, 1500000, by=500000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size=3) +
        geom_hline(yintercept=1085593, linetype="dashed", color = "red") #dashed line
is median SalePrice
n2 <- ggplot(data=dd, aes(x=Regionname)) +
        geom_histogram(stat='count', fill='#E64B35B2')+
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size=3)+
        theme(axis.text.x = element_text(angle=90, hjust = 1))
grid.arrange(n1, n2, nrow=1)
```





Features Engineering

- We convert Type, Method, CouncilArea and Regionname to factors
- We create a new variable “age”, which counts how many years from the house built to 2021.
- We divide Date into Year and Month to help us understand the relationship between Price and Date

```
dd$Type[dd$Type == 'h'] <- 1
dd$Type[dd$Type == 't'] <- 2
dd$Type[dd$Type == 'u'] <- 3
dd$Type <- as.factor(dd$Type)
```

```
dd$Method[dd$Method == 'PI'] <- 1
dd$Method[dd$Method == 'PN'] <- 2
dd$Method[dd$Method == 'S'] <- 3
dd$Method[dd$Method == 'SA'] <- 4
dd$Method[dd$Method == 'SN'] <- 5
dd$Method[dd$Method == 'SP'] <- 6
dd$Method[dd$Method == 'SS'] <- 7
dd$Method[dd$Method == 'VB'] <- 8
dd$Method[dd$Method == 'W'] <- 9
dd$Method <- as.factor(dd$Method)
```

```
dd$Regionname[dd$Regionname == 'Eastern Metropolitan'] <- 1
dd$Regionname[dd$Regionname == 'Eastern Victoria'] <- 2
dd$Regionname[dd$Regionname == 'Northern Metropolitan'] <- 3
dd$Regionname[dd$Regionname == 'Northern Victoria'] <- 4
dd$Regionname[dd$Regionname == 'South-Eastern Metropolitan'] <- 5
dd$Regionname[dd$Regionname == 'Southern Metropolitan'] <- 6
dd$Regionname[dd$Regionname == 'Western Metropolitan'] <- 7
dd$Regionname[dd$Regionname == 'Western Victoria'] <- 8
dd$Regionname <- as.factor(dd$Regionname)
```

```
dd$CouncilArea[dd$CouncilArea == 'Bayside City Council'] <- 1
dd$CouncilArea[dd$CouncilArea == 'Boroondara City Council'] <- 2
dd$CouncilArea[dd$CouncilArea == 'Brimbank City Council'] <- 3
dd$CouncilArea[dd$CouncilArea == 'Cardinia Shire Council'] <- 4
dd$CouncilArea[dd$CouncilArea == 'Casey City Council'] <- 5
dd$CouncilArea[dd$CouncilArea == 'Darebin City Council'] <- 6
dd$CouncilArea[dd$CouncilArea == 'Frankston City Council'] <- 7
dd$CouncilArea[dd$CouncilArea == 'Glen Eira City Council'] <- 8
dd$CouncilArea[dd$CouncilArea == 'Greater Dandenong City Council'] <- 9
dd$CouncilArea[dd$CouncilArea == 'Hobsons Bay City Council'] <- 10
dd$CouncilArea[dd$CouncilArea == 'Hume City Council'] <- 11
dd$CouncilArea[dd$CouncilArea == 'Kingston City Council'] <- 12
dd$CouncilArea[dd$CouncilArea == 'Knox City Council'] <- 13
dd$CouncilArea[dd$CouncilArea == 'Macedon Ranges Shire Council'] <- 14
dd$CouncilArea[dd$CouncilArea == 'Manningham City Council'] <- 15
dd$CouncilArea[dd$CouncilArea == 'Maribyrnong City Council'] <- 16
dd$CouncilArea[dd$CouncilArea == 'Maroondah City Council'] <- 17
dd$CouncilArea[dd$CouncilArea == 'Melbourne City Council'] <- 18
dd$CouncilArea[dd$CouncilArea == 'Melton City Council'] <- 19
dd$CouncilArea[dd$CouncilArea == 'Mitchell Shire Council'] <-20
dd$CouncilArea[dd$CouncilArea == 'Monash City Council'] <- 21
dd$CouncilArea[dd$CouncilArea == 'Moonee Valley City Council'] <- 22
dd$CouncilArea[dd$CouncilArea == 'Moorabool Shire Council'] <- 23
dd$CouncilArea[dd$CouncilArea == 'Moreland City Council'] <- 24
dd$CouncilArea[dd$CouncilArea == 'Nillumbik Shire Council'] <- 25
dd$CouncilArea[dd$CouncilArea == 'Port Phillip City Council'] <- 26
dd$CouncilArea[dd$CouncilArea == 'Stonnington City Council'] <- 27
dd$CouncilArea[dd$CouncilArea == 'Whitehorse City Council'] <- 28
dd$CouncilArea[dd$CouncilArea == 'Whittlesea City Council'] <- 29
dd$CouncilArea[dd$CouncilArea == 'Wyndham City Council'] <- 30
dd$CouncilArea[dd$CouncilArea == 'Yarra City Council'] <- 31
dd$CouncilArea[dd$CouncilArea == 'Yarra Ranges Shire Council'] <- 32
dd$CouncilArea[dd$CouncilArea == 'Banyule City Council'] <- 33
dd$CouncilArea <- as.factor(dd$CouncilArea)
```

```
dd$Date <- as.Date(dd$Date,"%d/%m/%Y")
dd$SellYear <- format(as.Date(dd$Date),"%Y")
dd$SellYear <- as.factor(dd$SellYear)
dd$SellMonth <- format(as.Date(dd$Date),"%m")
dd$SellMonth <- as.factor(dd$SellMonth)
```

```
dd <- dd[, age := 2021-YearBuilt]
```

```
dd <- dd[,-c('Suburb', 'Address', 'SellerG', 'Date', 'Postcode', 'YearBuilt', 'Lattit
ude', 'Longtitude')]
```

```
# standardize numeirc data
dd[] <- lapply(dd, function(x) if(is.numeric(x)){
                     scale(x, center=TRUE, scale=TRUE)
                      } else x)
```

```
str(dd)
```

```
## Classes 'data.table' and 'data.frame':   10613 obs. of  15 variables:
##  $ Rooms        : num  -1.0844 -0.0553 0.9739 -0.0553 -1.0844 ...
##   ..- attr(*, "scaled:center")= num 3.05
##   ..- attr(*, "scaled:scale")= num 0.972
##  $ Type         : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 3 1 1 ...
##  $ Price        : num  -0.0751 0.5635 0.764 1.1739 0.8175 ...
##   ..- attr(*, "scaled:center")= num 1085593
##   ..- attr(*, "scaled:scale")= num 673321
##  $ Method       : Factor w/ 5 levels "1","3","4","6",..: 2 4 5 2 2 2 5 2 2 2 ...
##  $ Distance     : num  -1.26 -1.26 -1.26 -1.26 -1.26 ...
##   ..- attr(*, "scaled:center")= num 11
##   ..- attr(*, "scaled:scale")= num 6.72
##  $ Bedroom2     : num  -1.066 -0.0363 -0.0363 0.9934 -1.066 ...
##   ..- attr(*, "scaled:center")= num 3.04
##   ..- attr(*, "scaled:scale")= num 0.971
##  $ Bathroom     : num  -0.876 0.528 -0.876 0.528 -0.876 ...
##   ..- attr(*, "scaled:center")= num 1.62
##   ..- attr(*, "scaled:scale")= num 0.712
##  $ Car          : num  -1.675 -1.675 0.352 -1.675 0.352 ...
##   ..- attr(*, "scaled:center")= num 1.65
##   ..- attr(*, "scaled:scale")= num 0.986
##  $ Landsize     : num  -0.338 -0.359 -0.372 -0.257 -0.246 ...
##   ..- attr(*, "scaled:center")= num 524
##   ..- attr(*, "scaled:scale")= num 1088
##  $ CouncilArea  : Factor w/ 33 levels "1","10","11",..: 25 25 25 25 25 25 25 25 25
25 ...
##  $ Regionname   : Factor w/ 8 levels "1","2","3","4",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ Propertycount: num  -0.798 -0.798 -0.798 -0.798 -0.798 ...
##   ..- attr(*, "scaled:center")= num 7498
##   ..- attr(*, "scaled:scale")= num 4359
##  $ SellYear     : Factor w/ 3 levels "2016","2017",..: 1 2 1 1 1 1 1 1 1 1 ...
##  $ SellMonth    : Factor w/ 12 levels "01","02","03",..: 2 3 6 5 10 10 11 11 10 7
...
##  $ age          : num  1.76 1.76 -1.32 1.49 2.03 ...
##   ..- attr(*, "scaled:center")= num 55.8
##   ..- attr(*, "scaled:scale")= num 37
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# split train/test sets
set.seed(810)
split = createDataPartition(y=dd$Price,p = 0.7,list = FALSE)
train = dd[split,]
test = dd[-split,]
y.train <- train$Price
y.test <- test$Price
```

Ridge and Lasso Regression

```
# formula
f1 <- as.formula(Price ~ Rooms + as.factor(Type) + as.factor(Method) + Distance + Bed
room2 + Bathroom + Car + Landsize + as.factor(CouncilArea)+ as.factor(Regionname) + a
s.factor(SellYear) +as.factor(SellMonth) + age + Propertycount)
```

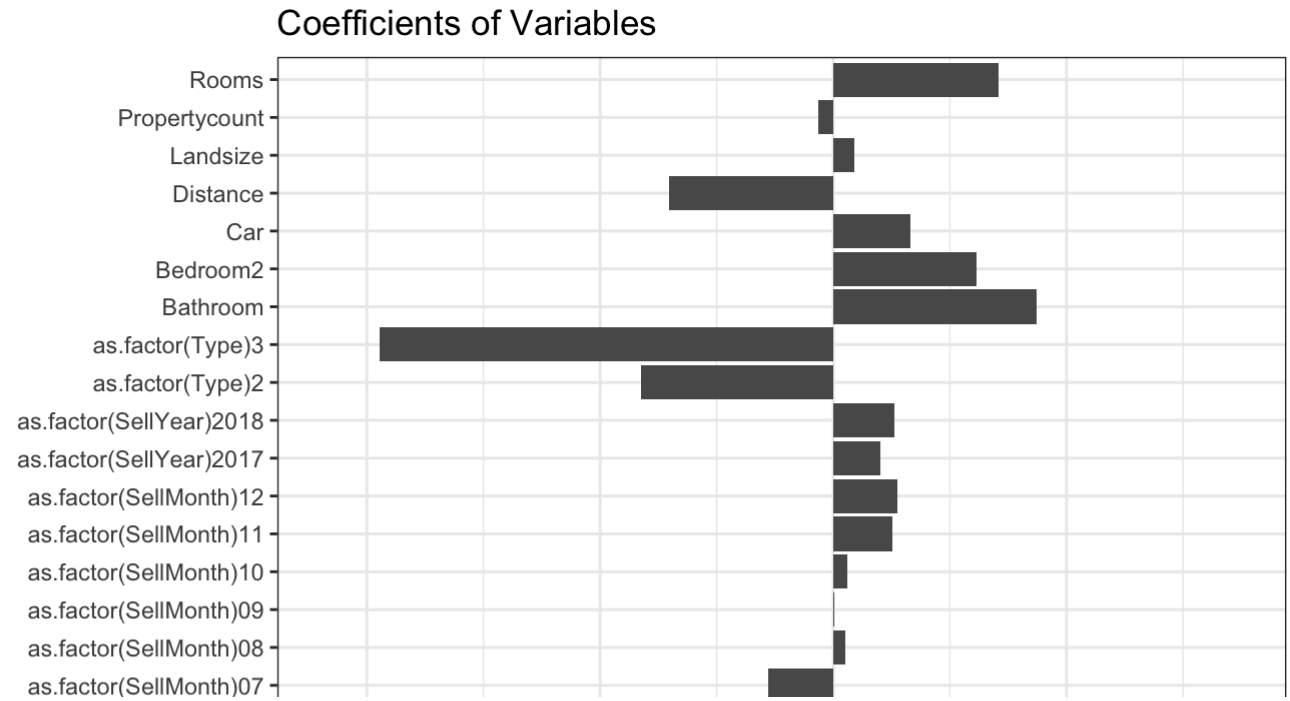
```
# creating matrix
x1.train <- model.matrix(f1, train)[, -1]
x1.test <- model.matrix(f1, test)[, -1]
```

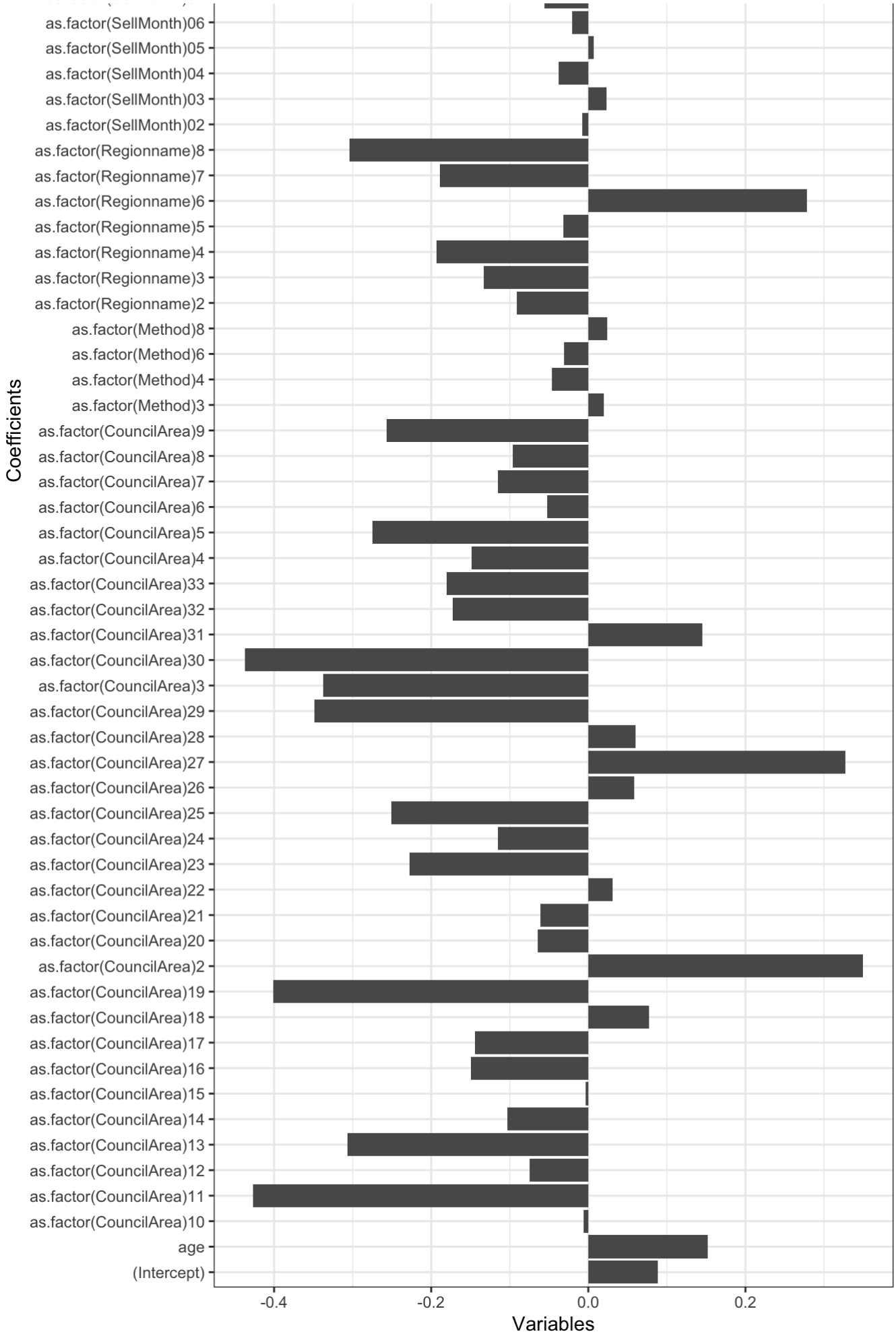
```
plot_coeffs <- function(model) {
  coeffs <- coefficients(model)
  df_coeffs <- data.frame("coeffs"=coeffs[,1],"variables"=rownames(coeffs))
  plot <-ggplot(df_coeffs, aes(x=variables, y=coeffs))+
  geom_bar(stat="identity") + labs(title = "Coefficients of Variables", x = "Coeffici
ents", y = "Variables")
  plot + coord_flip()
}
```

Ridge Regression with 10 fold cross validation

```
## [1] 0.3390804
```

```
## [1] 0.3671983
```




as.factor(SellMonth)06
as.factor(SellMonth)05
as.factor(SellMonth)04
as.factor(SellMonth)03
as.factor(SellMonth)02
as.factor(Regionname)8
as.factor(Regionname)7
as.factor(Regionname)6
as.factor(Regionname)5
as.factor(Regionname)4
as.factor(Regionname)3
as.factor(Regionname)2
as.factor(Method)8
as.factor(Method)6
as.factor(Method)4
as.factor(Method)3
as.factor(CouncilArea)9
as.factor(CouncilArea)8
as.factor(CouncilArea)7
as.factor(CouncilArea)6
as.factor(CouncilArea)5
as.factor(CouncilArea)4
as.factor(CouncilArea)33
as.factor(CouncilArea)32
as.factor(CouncilArea)31
as.factor(CouncilArea)30
as.factor(CouncilArea)3
as.factor(CouncilArea)29
as.factor(CouncilArea)28
as.factor(CouncilArea)27
as.factor(CouncilArea)26
as.factor(CouncilArea)25
as.factor(CouncilArea)24
as.factor(CouncilArea)23
as.factor(CouncilArea)22
as.factor(CouncilArea)21
as.factor(CouncilArea)20
as.factor(CouncilArea)2
as.factor(CouncilArea)19
as.factor(CouncilArea)18
as.factor(CouncilArea)17
as.factor(CouncilArea)16
as.factor(CouncilArea)15
as.factor(CouncilArea)14
as.factor(CouncilArea)13
as.factor(CouncilArea)12
as.factor(CouncilArea)11
as.factor(CouncilArea)10
age
(Intercept)
Coefficients
-0.4
-0.2
0.0
0.2
Variables

```
## 67 x 1 sparse Matrix of class "dgCMatrix"
##                                       1
## (Intercept)                0.0885489095
## Rooms                      0.1413302163
## as.factor(Type)2          -0.1646863390
## as.factor(Type)3          -0.3889798501
## as.factor(Method)3         0.0193292345
## as.factor(Method)4        -0.0461401549
## as.factor(Method)6        -0.0312332821
## as.factor(Method)8         0.0237282668
## Distance                  -0.1404842326
## Bedroom2                   0.1228675031
## Bathroom                   0.1739975145
## Car                        0.0657121216
## Landsize                   0.0183199689
## as.factor(CouncilArea)10 -0.0063368791
## as.factor(CouncilArea)11 -0.4264396980
## as.factor(CouncilArea)12 -0.0743932218
## as.factor(CouncilArea)13 -0.3062155905
## as.factor(CouncilArea)14 -0.1030543568
## as.factor(CouncilArea)15 -0.0031853911
## as.factor(CouncilArea)16 -0.1493195988
## as.factor(CouncilArea)17 -0.1439064424
## as.factor(CouncilArea)18  0.0776314295
## as.factor(CouncilArea)19 -0.4011367718
## as.factor(CouncilArea)2   0.3491963606
## as.factor(CouncilArea)20 -0.0640501426
## as.factor(CouncilArea)21 -0.0612726603
## as.factor(CouncilArea)22  0.0307948696
## as.factor(CouncilArea)23 -0.2279393389
## as.factor(CouncilArea)24 -0.1152278666
## as.factor(CouncilArea)25 -0.2504459045
## as.factor(CouncilArea)26  0.0583240661
## as.factor(CouncilArea)27  0.3274011469
## as.factor(CouncilArea)28  0.0602350357
## as.factor(CouncilArea)29 -0.3490114125
## as.factor(CouncilArea)3  -0.3375482499
## as.factor(CouncilArea)30 -0.4374967687
## as.factor(CouncilArea)31  0.1455080260
## as.factor(CouncilArea)32 -0.1726578730
## as.factor(CouncilArea)33 -0.1807513746
## as.factor(CouncilArea)4  -0.1488582312
## as.factor(CouncilArea)5  -0.2752327953
## as.factor(CouncilArea)6  -0.0527326247
## as.factor(CouncilArea)7  -0.1147557370
## as.factor(CouncilArea)8  -0.0960030445
## as.factor(CouncilArea)9  -0.2564650148
## as.factor(Regionname)2   -0.0914238814
## as.factor(Regionname)3   -0.1328068154
## as.factor(Regionname)4   -0.1934289091
## as.factor(Regionname)5   -0.0313718275
## as.factor(Regionname)6    0.2782996278
## as.factor(Regionname)7   -0.1885461440
## as.factor(Regionname)8   -0.3037516027
## as.factor(SellYear)2017   0.0403164389
## as.factor(SellYear)2018   0.0521920931
## as.factor(SellMonth)02   -0.0076203135
```

```
## as.factor(SellMonth)03    0.0233061104
## as.factor(SellMonth)04   -0.0377444231
## as.factor(SellMonth)05    0.0064920935
## as.factor(SellMonth)06   -0.0202957134
## as.factor(SellMonth)07   -0.0559699973
## as.factor(SellMonth)08    0.0101904166
## as.factor(SellMonth)09    0.0006672784
## as.factor(SellMonth)10    0.0123065900
## as.factor(SellMonth)11    0.0508105687
## as.factor(SellMonth)12    0.0548815789
## age                       0.1516394190
## Propertycount            -0.0127236594
```

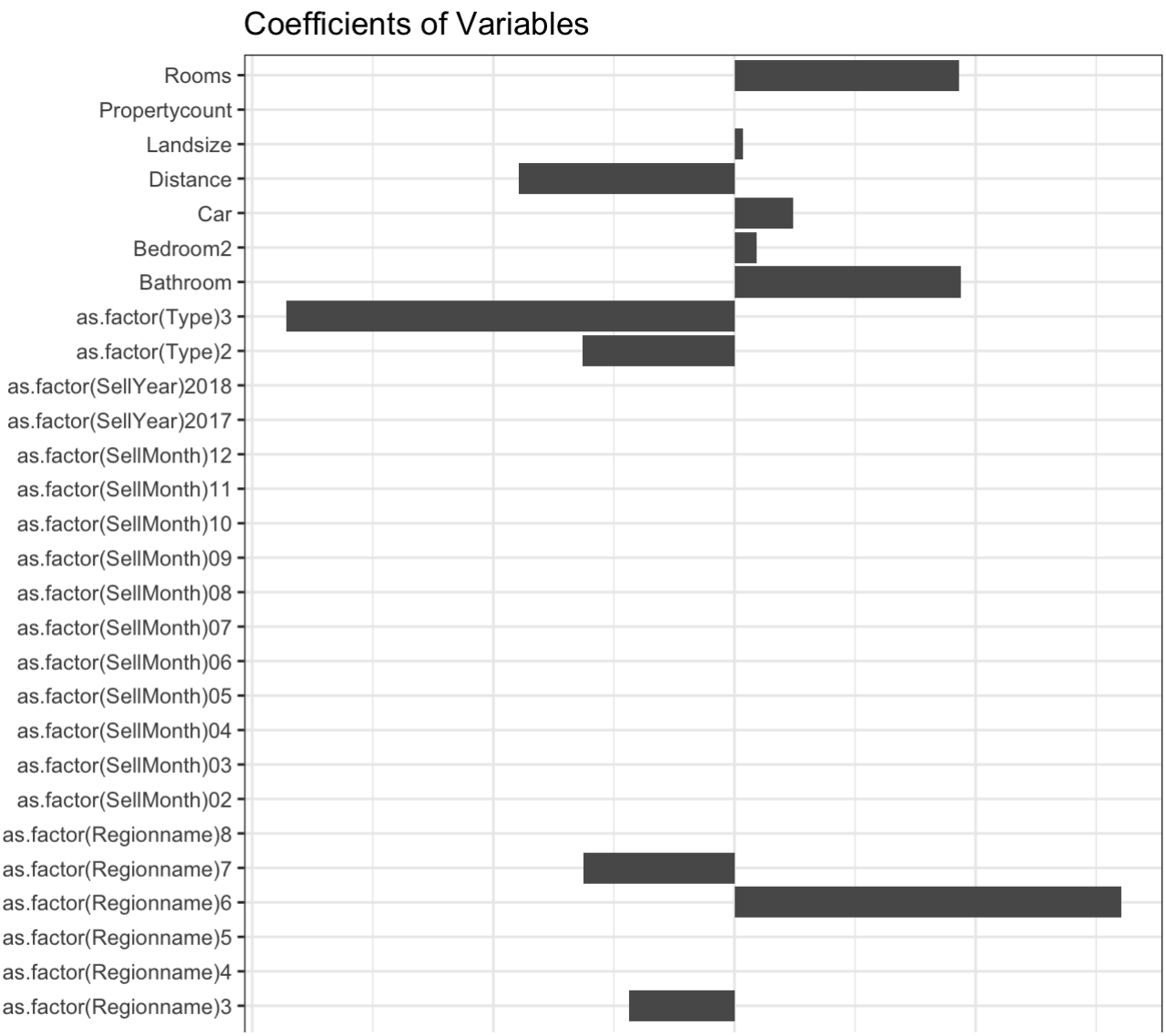
Results of Ridge

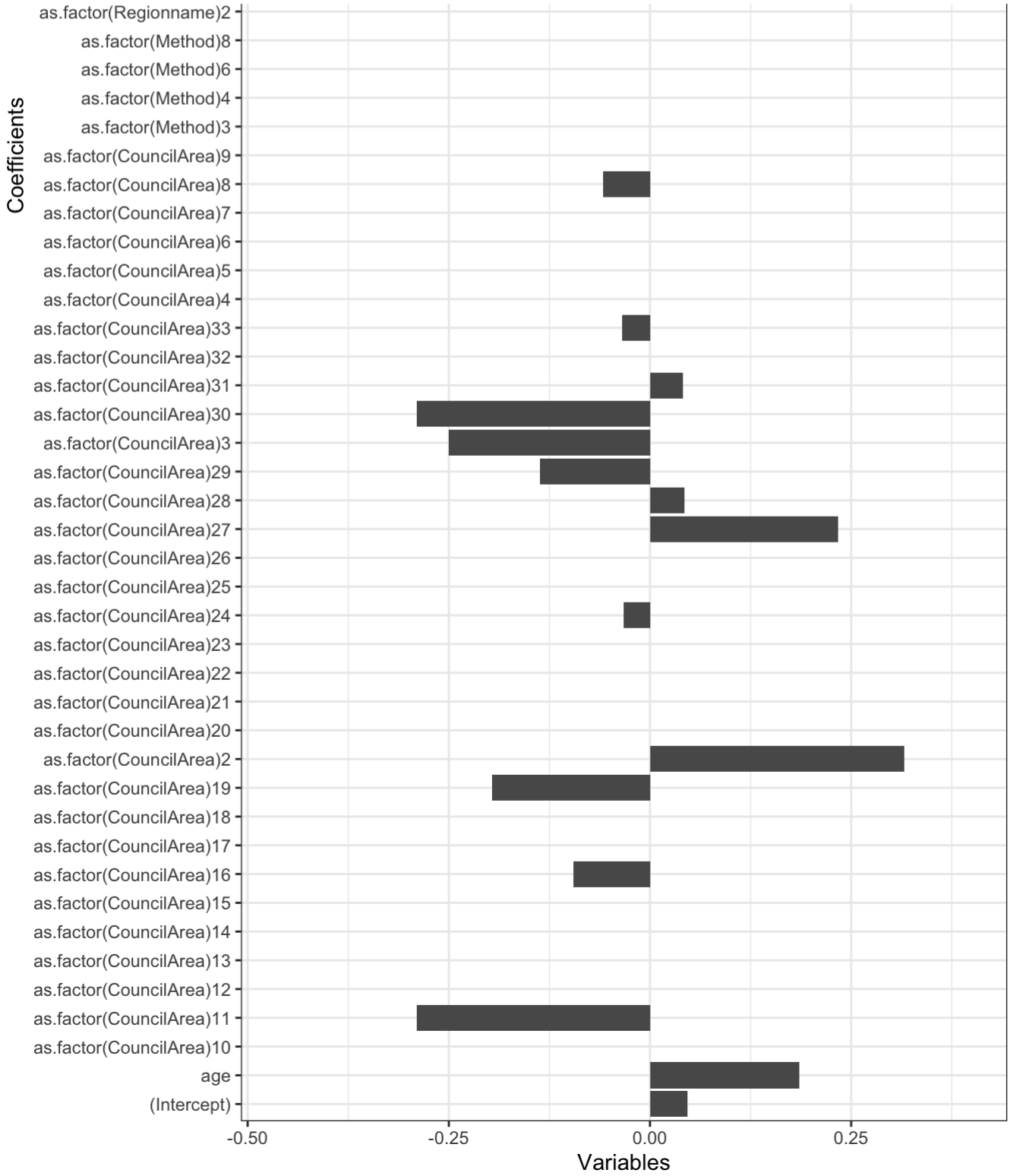
- For the ridge regression we did a 10 fold cross validation to find the lambda that best fit the model and data set.
- The train resulted in a MSE of 0.3353068 while test resulted in an expected higher MSE of 0.3611925.
- The CouncilArea and type of building seems to have a stronger effect on the price of a house.

Lasso Regression with 10 fold cross validation

```
## [1] 0.331511
```

```
## [1] 0.3563141
```




as.factor(Regionname)2
as.factor(Method)8
as.factor(Method)6
as.factor(Method)4
as.factor(Method)3
as.factor(CouncilArea)9
as.factor(CouncilArea)8
as.factor(CouncilArea)7
as.factor(CouncilArea)6
as.factor(CouncilArea)5
as.factor(CouncilArea)4
as.factor(CouncilArea)33
as.factor(CouncilArea)32
as.factor(CouncilArea)31
as.factor(CouncilArea)30
as.factor(CouncilArea)3
as.factor(CouncilArea)29
as.factor(CouncilArea)28
as.factor(CouncilArea)27
as.factor(CouncilArea)26
as.factor(CouncilArea)25
as.factor(CouncilArea)24
as.factor(CouncilArea)23
as.factor(CouncilArea)22
as.factor(CouncilArea)21
as.factor(CouncilArea)20
as.factor(CouncilArea)2
as.factor(CouncilArea)19
as.factor(CouncilArea)18
as.factor(CouncilArea)17
as.factor(CouncilArea)16
as.factor(CouncilArea)15
as.factor(CouncilArea)14
as.factor(CouncilArea)13
as.factor(CouncilArea)12
as.factor(CouncilArea)11
as.factor(CouncilArea)10
age
(Intercept)
Coefficients
-0.50
-0.25
0.00
0.25
Variables

```
## 67 x 1 sparse Matrix of class "dgCMatrix"
##                                    1
## (Intercept)                0.04703541
## Rooms                      0.23325818
## as.factor(Type)2          -0.15747812
## as.factor(Type)3          -0.46505897
## as.factor(Method)3         .
## as.factor(Method)4         .
## as.factor(Method)6         .
## as.factor(Method)8         .
## Distance                  -0.22369937
## Bedroom2                   0.02336924
## Bathroom                   0.23506607
## Car                        0.06081136
## Landsize                   0.00837035
## as.factor(CouncilArea)10   .
## as.factor(CouncilArea)11  -0.29015230
## as.factor(CouncilArea)12   .
## as.factor(CouncilArea)13   .
## as.factor(CouncilArea)14   .
## as.factor(CouncilArea)15   .
## as.factor(CouncilArea)16  -0.09511002
## as.factor(CouncilArea)17   .
## as.factor(CouncilArea)18   .
## as.factor(CouncilArea)19  -0.19628882
## as.factor(CouncilArea)2    0.31623971
## as.factor(CouncilArea)20   .
## as.factor(CouncilArea)21   .
## as.factor(CouncilArea)22   .
## as.factor(CouncilArea)23   .
## as.factor(CouncilArea)24  -0.03284146
## as.factor(CouncilArea)25   .
## as.factor(CouncilArea)26   .
## as.factor(CouncilArea)27   0.23422970
## as.factor(CouncilArea)28   0.04283804
## as.factor(CouncilArea)29  -0.13625294
## as.factor(CouncilArea)3   -0.25013465
## as.factor(CouncilArea)30  -0.28944638
## as.factor(CouncilArea)31   0.04062636
## as.factor(CouncilArea)32   .
## as.factor(CouncilArea)33  -0.03420086
## as.factor(CouncilArea)4    .
## as.factor(CouncilArea)5    .
## as.factor(CouncilArea)6    .
## as.factor(CouncilArea)7    .
## as.factor(CouncilArea)8   -0.05824913
## as.factor(CouncilArea)9    .
## as.factor(Regionname)2     .
## as.factor(Regionname)3    -0.10947424
## as.factor(Regionname)4     .
## as.factor(Regionname)5     .
## as.factor(Regionname)6     0.40067834
## as.factor(Regionname)7    -0.15617589
## as.factor(Regionname)8     .
## as.factor(SellYear)2017    .
## as.factor(SellYear)2018    .
## as.factor(SellMonth)02     .
```

```
## as.factor(SellMonth)03    .
## as.factor(SellMonth)04    .
## as.factor(SellMonth)05    .
## as.factor(SellMonth)06    .
## as.factor(SellMonth)07    .
## as.factor(SellMonth)08    .
## as.factor(SellMonth)09    .
## as.factor(SellMonth)10    .
## as.factor(SellMonth)11    .
## as.factor(SellMonth)12    .
## age                      0.18546813
## Propertycount            .
```

Result of Lasso

- For the lasso regression we did a 10 fold cross validation to find the lambda that best fit the model and data set.
- The train resulted in a MSE of 0.3316269 while test resulted in an expected higher MSE of 0.3562078.
- Again the CouncilArea and type of building seems to have a stronger effect on the price of a house. Specifically Boroondara City Council(0.412293150) and Type3(-0.451226392).

Overall Result

Over all the redeuced dimension lasso regression did better than the ridge. The train for ridge and lasso were respectively 0.3353068(ridge) and 0.3316269(lasso). The test for ridge and lasso were respectively 0.3611925(ridge) and 0.3562078 (lasso).

Linear Regression

Using all variables to get the full model

```
model1 <- lm(Price ~ Rooms+factor(Type)+factor(Method)+Distance+Bedroom2+Bathroom+Car
+Landsize+age+factor(CouncilArea)+factor(Regionname)+Propertycount+factor(SellYear)+f
actor(SellMonth),data = train)
summary(model1)
```

```
##
## Call:
## lm(formula = Price ~ Rooms + factor(Type) + factor(Method) +
##     Distance + Bedroom2 + Bathroom + Car + Landsize + age + factor(CouncilArea) +
##     factor(Regionname) + Propertycount + factor(SellYear) + factor(SellMonth),
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3206 -0.3025 -0.0465  0.2192 12.0922
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.8482788  0.0966550   8.776  < 2e-16 ***
## Rooms                   0.2425881  0.0263545   9.205  < 2e-16 ***
## factor(Type)2          -0.3053318  0.0283716 -10.762  < 2e-16 ***
## factor(Type)3          -0.6000913  0.0242691 -24.727  < 2e-16 ***
## factor(Method)3         0.0768356  0.0212910   3.609 0.000310 ***
## factor(Method)4         0.0064820  0.0805720   0.080 0.935882
## factor(Method)6         0.0363164  0.0267660   1.357 0.174884
## factor(Method)8         0.0106145  0.0291563   0.364 0.715827
## Distance               -0.3499170  0.0188180 -18.595  < 2e-16 ***
## Bedroom2                0.0129393  0.0262245   0.493 0.621741
## Bathroom                0.2256208  0.0093369  24.165  < 2e-16 ***
## Car                     0.0817031  0.0076191  10.724  < 2e-16 ***
## Landsize                0.0248491  0.0067591   3.676 0.000238 ***
## age                     0.1489527  0.0091112  16.348  < 2e-16 ***
## factor(CouncilArea)10 -0.9955253  0.1047841  -9.501  < 2e-16 ***
## factor(CouncilArea)11 -1.1718454  0.0687235 -17.052  < 2e-16 ***
## factor(CouncilArea)12 -0.6695824  0.0804952  -8.318  < 2e-16 ***
## factor(CouncilArea)13 -1.1230868  0.0901663 -12.456  < 2e-16 ***
## factor(CouncilArea)14 -0.6775544  0.2768436  -2.447 0.014411 *
## factor(CouncilArea)15 -0.8828966  0.0757379 -11.657  < 2e-16 ***
## factor(CouncilArea)16 -1.2420941  0.1042269 -11.917  < 2e-16 ***
## factor(CouncilArea)17 -0.8642469  0.0908742  -9.510  < 2e-16 ***
## factor(CouncilArea)18 -0.6313204  0.0592043 -10.663  < 2e-16 ***
## factor(CouncilArea)19 -1.4674512  0.1335898 -10.985  < 2e-16 ***
## factor(CouncilArea)2  -0.0817825  0.0412653  -1.982 0.047531 *
## factor(CouncilArea)20 -0.5250991  0.4459976  -1.177 0.239090
## factor(CouncilArea)21 -0.6774919  0.0535364 -12.655  < 2e-16 ***
## factor(CouncilArea)22 -0.9571463  0.1024837  -9.339  < 2e-16 ***
## factor(CouncilArea)23 -1.0777080  0.6024503  -1.789 0.073676 .
## factor(CouncilArea)24 -0.9352782  0.0668556 -13.990  < 2e-16 ***
## factor(CouncilArea)25 -1.4207950  0.2008197  -7.075 1.63e-12 ***
## factor(CouncilArea)26 -0.4669467  0.0509231  -9.170  < 2e-16 ***
## factor(CouncilArea)27 -0.0799081  0.0517162  -1.545 0.122358
## factor(CouncilArea)28 -0.7349315  0.0847724  -8.669  < 2e-16 ***
## factor(CouncilArea)29 -1.0197528  0.0735415 -13.866  < 2e-16 ***
## factor(CouncilArea)3  -1.3595543  0.0998135 -13.621  < 2e-16 ***
## factor(CouncilArea)30 -1.4828081  0.1074022 -13.806  < 2e-16 ***
## factor(CouncilArea)31 -0.7060153  0.0763485  -9.247  < 2e-16 ***
## factor(CouncilArea)32 -0.8714447  0.2169479  -4.017 5.96e-05 ***
## factor(CouncilArea)33 -1.0881781  0.0657603 -16.548  < 2e-16 ***
## factor(CouncilArea)4  -0.4065438  0.2508821  -1.620 0.105177
## factor(CouncilArea)5  -0.7981779  0.1359495  -5.871 4.52e-09 ***
## factor(CouncilArea)6  -0.8456691  0.0675718 -12.515  < 2e-16 ***
## factor(CouncilArea)7  -0.3563268  0.1189621  -2.995 0.002751 **
```

```
## factor(CouncilArea)8  -0.5336353  0.0422552 -12.629  < 2e-16 ***
## factor(CouncilArea)9  -0.9115132  0.1237168  -7.368 1.92e-13 ***
## factor(Regionname)2    0.0905513  0.1707853   0.530 0.595986
## factor(Regionname)3   -0.3513316  0.0541962  -6.483 9.60e-11 ***
## factor(Regionname)4    0.3194124  0.1783884   1.791 0.073408 .
## factor(Regionname)5   -0.0601803  0.0824464  -0.730 0.465455
## factor(Regionname)6   -0.1811291  0.0557133  -3.251 0.001155 **
## factor(Regionname)7   -0.1783198  0.0910569  -1.958 0.050229 .
## factor(Regionname)8    0.2464949  0.1740447   1.416 0.156738
## Propertycount          0.0001572  0.0082931   0.019 0.984874
## factor(SellYear)2017   0.1063480  0.0166304   6.395 1.71e-10 ***
## factor(SellYear)2018   0.2019621  0.0409390   4.933 8.26e-07 ***
## factor(SellMonth)02    0.0435521  0.0623975   0.698 0.485213
## factor(SellMonth)03    0.0882784  0.0622399   1.418 0.156129
## factor(SellMonth)04    0.0713149  0.0721894   0.988 0.323241
## factor(SellMonth)05    0.1266286  0.0692391   1.829 0.067461 .
## factor(SellMonth)06    0.0976808  0.0694684   1.406 0.159731
## factor(SellMonth)07    0.0925044  0.0689839   1.341 0.179976
## factor(SellMonth)08    0.1183329  0.0701300   1.687 0.091581 .
## factor(SellMonth)09    0.1238019  0.0697309   1.775 0.075869 .
## factor(SellMonth)10    0.1407462  0.0697173   2.019 0.043543 *
## factor(SellMonth)11    0.1935025  0.0702731   2.754 0.005909 **
## factor(SellMonth)12    0.2107898  0.0725263   2.906 0.003667 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5783 on 7364 degrees of freedom
## Multiple R-squared:  0.6633, Adjusted R-squared:  0.6603
## F-statistic: 219.8 on 66 and 7364 DF,  p-value: < 2.2e-16
```

```
yhat.train.m1 <- predict(model1)
mse.train.m1 <- mean((y.train - yhat.train.m1)^2)
mse.train.m1
```

```
## [1] 0.3314157
```

```
yhat.test.m1 <- predict(model1, test)
mse.test.m1 <- mean((y.test - yhat.test.m1)^2)
mse.test.m1
```

```
## [1] 0.3560723
```

Using significant variables

```
model2 <- lm(Price ~ Rooms+factor(Type)+Distance+Bathroom+Car+Landsize+age+factor(Cou
ncilArea)+factor(SellYear),data = train)
summary(model2)
```

```
##
## Call:
## lm(formula = Price ~ Rooms + factor(Type) + Distance + Bathroom +
##     Car + Landsize + age + factor(CouncilArea) + factor(SellYear),
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3450 -0.3023 -0.0524  0.2228 12.0448
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.857542   0.034652  24.747  < 2e-16 ***
## Rooms                    0.254869   0.010729  23.754  < 2e-16 ***
## factor(Type)2           -0.308873   0.028401 -10.875  < 2e-16 ***
## factor(Type)3           -0.604018   0.024120 -25.042  < 2e-16 ***
## Distance                -0.319276   0.016996 -18.785  < 2e-16 ***
## Bathroom                 0.227064   0.009285  24.454  < 2e-16 ***
## Car                      0.082311   0.007637  10.779  < 2e-16 ***
## Landsize                 0.026764   0.006781   3.947 7.99e-05 ***
## age                      0.154994   0.009110  17.014  < 2e-16 ***
## factor(CouncilArea)10 -0.981547   0.050493 -19.439  < 2e-16 ***
## factor(CouncilArea)11 -1.314653   0.048579 -27.062  < 2e-16 ***
## factor(CouncilArea)12 -0.594632   0.057460 -10.349  < 2e-16 ***
## factor(CouncilArea)13 -0.970444   0.075015 -12.937  < 2e-16 ***
## factor(CouncilArea)14 -0.344867   0.229487  -1.503  0.13294
## factor(CouncilArea)15 -0.706206   0.051413 -13.736  < 2e-16 ***
## factor(CouncilArea)16 -1.222779   0.046333 -26.391  < 2e-16 ***
## factor(CouncilArea)17 -0.705789   0.071974  -9.806  < 2e-16 ***
## factor(CouncilArea)18 -0.686166   0.049699 -13.806  < 2e-16 ***
## factor(CouncilArea)19 -1.218817   0.080471 -15.146  < 2e-16 ***
## factor(CouncilArea)2  -0.062077   0.040834  -1.520  0.12850
## factor(CouncilArea)20 -0.096430   0.419841  -0.230  0.81834
## factor(CouncilArea)21 -0.611535   0.050086 -12.210  < 2e-16 ***
## factor(CouncilArea)22 -0.936775   0.043855 -21.360  < 2e-16 ***
## factor(CouncilArea)23 -0.710677   0.585676  -1.213  0.22500
## factor(CouncilArea)24 -1.081536   0.043233 -25.016  < 2e-16 ***
## factor(CouncilArea)25 -0.988716   0.140209  -7.052 1.93e-12 ***
## factor(CouncilArea)26 -0.447307   0.050254  -8.901  < 2e-16 ***
## factor(CouncilArea)27 -0.060337   0.051447  -1.173  0.24092
## factor(CouncilArea)28 -0.547525   0.064150  -8.535  < 2e-16 ***
## factor(CouncilArea)29 -1.188238   0.056039 -21.204  < 2e-16 ***
## factor(CouncilArea)3  -1.351195   0.045641 -29.605  < 2e-16 ***
## factor(CouncilArea)30 -1.470138   0.061594 -23.868  < 2e-16 ***
## factor(CouncilArea)31 -0.832112   0.054270 -15.333  < 2e-16 ***
## factor(CouncilArea)32 -0.648251   0.154386  -4.199 2.71e-05 ***
## factor(CouncilArea)33 -0.995470   0.044477 -22.382  < 2e-16 ***
## factor(CouncilArea)4  -0.244988   0.194213  -1.261  0.20719
## factor(CouncilArea)5  -0.722268   0.114661  -6.299 3.16e-10 ***
## factor(CouncilArea)6  -1.001164   0.041469 -24.142  < 2e-16 ***
## factor(CouncilArea)7  -0.310886   0.098102  -3.169  0.00154 **
## factor(CouncilArea)8  -0.533241   0.042388 -12.580  < 2e-16 ***
## factor(CouncilArea)9  -0.825871   0.103339  -7.992 1.53e-15 ***
## factor(SellYear)2017   0.075682   0.015291   4.949 7.61e-07 ***
## factor(SellYear)2018   0.103583   0.024545   4.220 2.47e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.5816 on 7388 degrees of freedom
## Multiple R-squared:  0.6583, Adjusted R-squared:  0.6564
## F-statistic:   339 on 42 and 7388 DF,  p-value: < 2.2e-16
```

```
yhat.train.m2 <- predict(model2)
mse.train.m2 <- mean((y.train - yhat.train.m2)^2)
mse.train.m2
```

```
## [1] 0.3362878
```

```
yhat.test.m2 <- predict(model2, test)
mse.test.m2 <- mean((y.test - yhat.test.m2)^2)
mse.test.m2
```

```
## [1] 0.3626219
```

Reduce to more variables

```
model3 <- lm(Price ~ Rooms+factor(Type)+Distance+Bathroom+Car+Landsize+age+factor(Sel
lYear),data = train)
summary(model3)
```

```
##
## Call:
## lm(formula = Price ~ Rooms + factor(Type) + Distance + Bathroom +
##     Car + Landsize + age + factor(SellYear), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6800 -0.3908 -0.0923  0.2723 12.3541
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.043709   0.015425   2.834  0.00461 **
## Rooms                 0.254513   0.012659  20.105  < 2e-16 ***
## factor(Type)2        -0.105275   0.033452  -3.147  0.00166 **
## factor(Type)3        -0.268335   0.027808  -9.650  < 2e-16 ***
## Distance             -0.303465   0.009642 -31.474  < 2e-16 ***
## Bathroom              0.332605   0.010835  30.698  < 2e-16 ***
## Car                   0.095481   0.009061  10.537  < 2e-16 ***
## Landsize              0.033906   0.007452   4.550 5.45e-06 ***
## age                   0.285957   0.010333  27.673  < 2e-16 ***
## factor(SellYear)2017  0.031465   0.018063   1.742  0.08156 .
## factor(SellYear)2018 -0.005209   0.028863  -0.180  0.85680
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6981 on 7420 degrees of freedom
## Multiple R-squared:  0.5057, Adjusted R-squared:  0.505
## F-statistic:   759 on 10 and 7420 DF,  p-value: < 2.2e-16
```

```
yhat.train.m3 <- predict(model3)
mse.train.m3 <- mean((y.train - yhat.train.m3)^2)
mse.train.m3
```

```
## [1] 0.4865665
```

```
yhat.test.m3 <- predict(model3, test)
mse.test.m3 <- mean((y.test - yhat.test.m3)^2)
mse.test.m3
```

```
## [1] 0.5138
```

Variable interaction model

```
model4 <- lm(Price ~ Rooms+factor(Type)+Distance+Bathroom+Car+Landsize+age+Landsize*a
ge+factor(SellYear)+factor(CouncilArea),data = train)
summary(model4)
```

```
##
## Call:
## lm(formula = Price ~ Rooms + factor(Type) + Distance + Bathroom +
##     Car + Landsize + age + Landsize * age + factor(SellYear) +
##     factor(CouncilArea), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4044 -0.2997 -0.0511  0.2220 12.0344
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.858960   0.034550  24.861  < 2e-16 ***
## Rooms                   0.251438   0.010710  23.477  < 2e-16 ***
## factor(Type)2          -0.318607   0.028355 -11.237  < 2e-16 ***
## factor(Type)3          -0.605944   0.024050 -25.195  < 2e-16 ***
## Distance               -0.321970   0.016951 -18.994  < 2e-16 ***
## Bathroom                0.228179   0.009259  24.643  < 2e-16 ***
## Car                     0.077861   0.007643  10.187  < 2e-16 ***
## Landsize                0.071337   0.009495   7.513 6.44e-14 ***
## age                     0.157917   0.009093  17.366  < 2e-16 ***
## factor(SellYear)2017    0.075548   0.015246   4.955 7.38e-07 ***
## factor(SellYear)2018    0.106179   0.024476   4.338 1.46e-05 ***
## factor(CouncilArea)10 -0.977090   0.050348 -19.407  < 2e-16 ***
## factor(CouncilArea)11 -1.310815   0.048440 -27.061  < 2e-16 ***
## factor(CouncilArea)12 -0.593636   0.057291 -10.362  < 2e-16 ***
## factor(CouncilArea)13 -0.966744   0.074797 -12.925  < 2e-16 ***
## factor(CouncilArea)14 -0.434226   0.229201  -1.895  0.05820 .
## factor(CouncilArea)15 -0.706234   0.051262 -13.777  < 2e-16 ***
## factor(CouncilArea)16 -1.215910   0.046208 -26.314  < 2e-16 ***
## factor(CouncilArea)17 -0.706137   0.071762  -9.840  < 2e-16 ***
## factor(CouncilArea)18 -0.679897   0.049561 -13.718  < 2e-16 ***
## factor(CouncilArea)19 -1.209892   0.080245 -15.077  < 2e-16 ***
## factor(CouncilArea)2  -0.067744   0.040722  -1.664  0.09625 .
## factor(CouncilArea)20 -0.083666   0.418609  -0.200  0.84159
## factor(CouncilArea)21 -0.612506   0.049938 -12.265  < 2e-16 ***
## factor(CouncilArea)22 -0.934837   0.043727 -21.379  < 2e-16 ***
## factor(CouncilArea)23 -0.699845   0.583954  -1.198  0.23078
## factor(CouncilArea)24 -1.076740   0.043112 -24.976  < 2e-16 ***
## factor(CouncilArea)25 -0.987348   0.139796  -7.063 1.78e-12 ***
## factor(CouncilArea)26 -0.436845   0.050130  -8.714  < 2e-16 ***
## factor(CouncilArea)27 -0.058241   0.051297  -1.135  0.25626
## factor(CouncilArea)28 -0.551777   0.063965  -8.626  < 2e-16 ***
## factor(CouncilArea)29 -1.183735   0.055878 -21.184  < 2e-16 ***
## factor(CouncilArea)3  -1.349200   0.045508 -29.648  < 2e-16 ***
## factor(CouncilArea)30 -1.466732   0.061415 -23.882  < 2e-16 ***
## factor(CouncilArea)31 -0.822455   0.054130 -15.194  < 2e-16 ***
## factor(CouncilArea)32 -0.651686   0.153932  -4.234 2.33e-05 ***
## factor(CouncilArea)33 -0.998527   0.044348 -22.516  < 2e-16 ***
## factor(CouncilArea)4  -0.231069   0.193652  -1.193  0.23282
## factor(CouncilArea)5  -0.710946   0.114336  -6.218 5.31e-10 ***
## factor(CouncilArea)6  -0.999630   0.041348 -24.176  < 2e-16 ***
## factor(CouncilArea)7  -0.299065   0.097829  -3.057  0.00224 **
## factor(CouncilArea)8  -0.531897   0.042264 -12.585  < 2e-16 ***
## factor(CouncilArea)9  -0.820945   0.103037  -7.967 1.86e-15 ***
## Landsize:age            0.059548   0.008906   6.686 2.45e-11 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5799 on 7387 degrees of freedom
## Multiple R-squared:  0.6604, Adjusted R-squared:  0.6584
## F-statistic: 334.1 on 43 and 7387 DF,  p-value: < 2.2e-16
```

```
yhat.train.m4 <- predict(model4)
mse.train.m4 <- mean((y.train - yhat.train.m4)^2)
mse.train.m4
```

```
## [1] 0.3342648
```

```
yhat.test.m4 <- predict(model4, test)
mse.test.m4 <- mean((y.test - yhat.test.m4)^2)
mse.test.m4
```

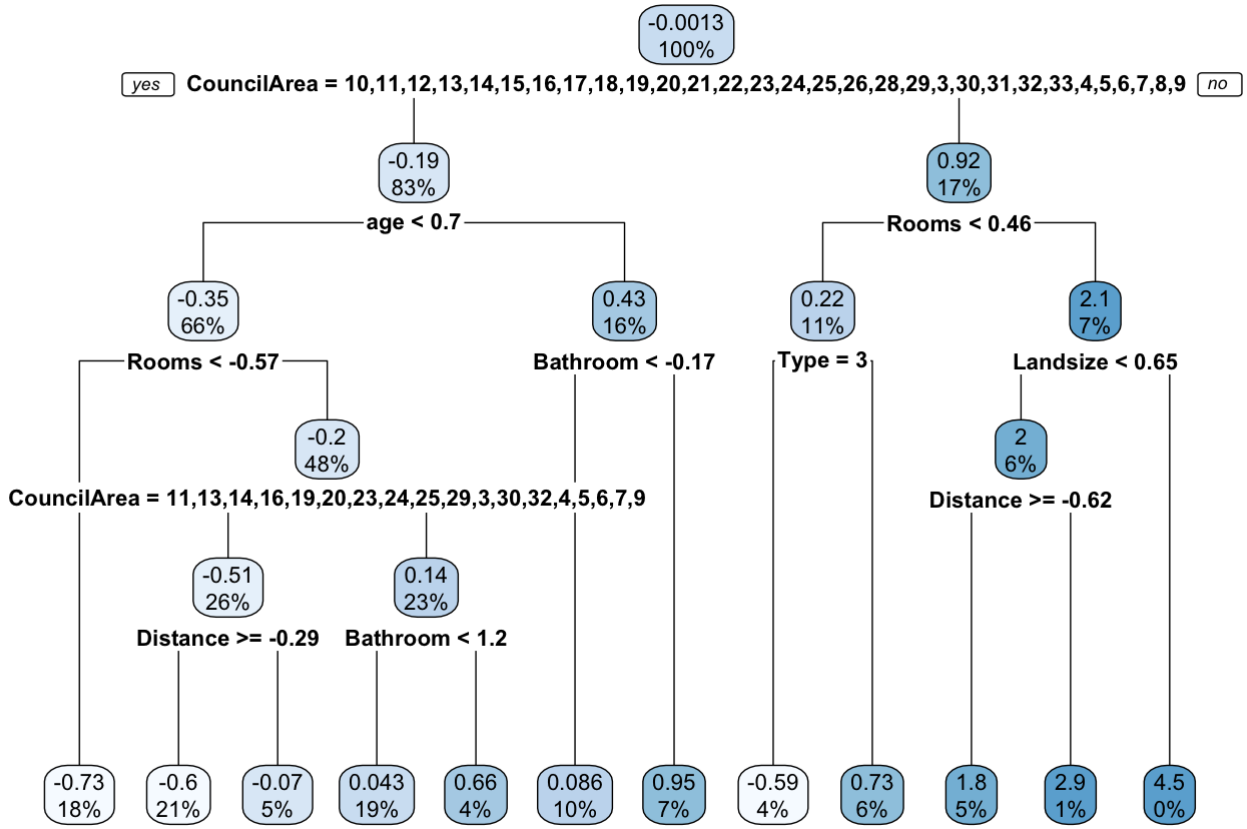
```
## [1] 0.3606024
```

Overall Result

The full model has the lowest test MSE. The interaction model also performed well. There might be more complex relationship between housing price and other variables.

Regression Trees

```
RT_fit <- rpart(Price~., data = train, method = 'anova')
#we specify anova as we aim for regression (rather than classification)
rpart.plot(RT_fit)
```




```
predictions1 <- predict(RT_fit,train)
Rmse1 <- rmse(predictions1, y.train)
MSE1_train=Rmse1^2
print(MSE1_train)
```

```
## [1] 0.3827541
```

```
predictions2 <- predict(RT_fit, test)
Rmse2 <- rmse(predictions2, y.test)
MSE1_test=Rmse2^2
print(MSE1_test)
```

```
## [1] 0.4011495
```

```
summary(RT_fit)
```

```
## Call:
## rpart(formula = Price ~ ., data = train, method = "anova")
##   n= 7431
##
##            CP nsplit rel error    xerror       xstd
## 1  0.17763964      0 1.0000000 1.0000879 0.04183618
## 2  0.13857705      1 0.8223604 0.8231385 0.03588324
## 3  0.08153327      2 0.6837833 0.6846174 0.03179165
## 4  0.04411059      3 0.6022501 0.6058084 0.03073854
## 5  0.04406257      5 0.5140289 0.5730886 0.03062170
## 6  0.03011475      6 0.4699663 0.4766117 0.02882395
## 7  0.01667890      7 0.4398515 0.4461393 0.02793310
## 8  0.01158270      8 0.4231726 0.4505378 0.02813125
## 9  0.01157281      9 0.4115900 0.4405999 0.02733548
## 10 0.01115889     10 0.4000171 0.4356354 0.02738020
## 11 0.01000000     11 0.3888583 0.4227077 0.02717378
##
## Variable importance
##   CouncilArea         Rooms      Bedroom2      Landsize           age
##            20            18            17             9             8
##          Type      Bathroom           Car    Regionname      Distance
##             8             8             4             4             3
## Propertycount
##             1
##
## Node number 1: 7431 observations,    complexity param=0.1776396
##   mean=-0.001306507, MSE=0.9843023
##   left son=2 (6164 obs) right son=3 (1267 obs)
##   Primary splits:
##       CouncilArea splits as  RLLLLLLLLLLRLLLLLLLRLLLLLLLLLLLL, improve=0.1776396,
(0 missing)
##       Rooms       < 0.459301    to the left,  improve=0.1614376, (0 missing)
##       Bedroom2    < 0.4785597   to the left,  improve=0.1562118, (0 missing)
##       age         < 0.4256092   to the left,  improve=0.1535663, (0 missing)
##       Bathroom    < 1.230347    to the left,  improve=0.1480135, (0 missing)
##   Surrogate splits:
##       Regionname splits as  LLLLLRLL, agree=0.847, adj=0.103, (0 split)
##
## Node number 2: 6164 observations,    complexity param=0.08153327
##   mean=-0.1908859, MSE=0.5568506
##   left son=4 (4941 obs) right son=5 (1223 obs)
##   Primary splits:
##       age         < 0.6960124   to the left,  improve=0.1737438, (0 missing)
##       Type        splits as  RLL, improve=0.1313282, (0 missing)
##       Rooms       < 0.459301    to the left,  improve=0.1107988, (0 missing)
##       Bedroom2    < -0.5511319  to the left,  improve=0.1050146, (0 missing)
##       CouncilArea splits as  -RLRLLRLRRL-LRRLLLR-RLLLRLRLLRLRL, improve=0.1010791,
(0 missing)
##   Surrogate splits:
##       Car < -1.168103   to the right, agree=0.831, adj=0.15, (0 split)
##
## Node number 3: 1267 observations,    complexity param=0.138577
##   mean=0.9210041, MSE=2.038362
##   left son=6 (781 obs) right son=7 (486 obs)
##   Primary splits:
##       Rooms    < 0.459301    to the left,  improve=0.3924724, (0 missing)
##       Bedroom2 < 0.4785597   to the left,  improve=0.3849063, (0 missing)
```

```
##       Type     splits as  RLL, improve=0.3746916, (0 missing)
##       Bathroom < -0.1740814  to the left,  improve=0.3110188, (0 missing)
##       Landsize < -0.1540783  to the left,  improve=0.2857763, (0 missing)
##   Surrogate splits:
##       Bedroom2 < 0.4785597   to the left,  agree=0.976, adj=0.936, (0 split)
##       Landsize < -0.04099314 to the left,  agree=0.766, adj=0.389, (0 split)
##       Bathroom < 1.230347    to the left,  agree=0.756, adj=0.364, (0 split)
##       Car      < -0.1544004  to the left,  agree=0.687, adj=0.185, (0 split)
##       Type     splits as  RLL, agree=0.681, adj=0.169, (0 split)
##
## Node number 4: 4941 observations,    complexity param=0.04411059
##   mean=-0.3456356, MSE=0.3937709
##   left son=8 (1340 obs) right son=9 (3601 obs)
##   Primary splits:
##       Rooms       < -0.5698475  to the left,  improve=0.1397200, (0 missing)
##       Bedroom2    < -0.5511319  to the left,  improve=0.1344693, (0 missing)
##       Type        splits as  RRL, improve=0.1281932, (0 missing)
##       CouncilArea splits as  -RLRLLRLRLL-LRRLLLL-RLLLLLRLLLLRL, improve=0.1273898,
(0 missing)
##       Bathroom    < 1.230347    to the left,  improve=0.1197290, (0 missing)
##   Surrogate splits:
##       Bedroom2    < -0.5511319  to the left,  agree=0.989, adj=0.960, (0 split)
##       Type        splits as  RRL, agree=0.880, adj=0.559, (0 split)
##       Landsize    < -0.3186494  to the left,  agree=0.844, adj=0.424, (0 split)
##       CouncilArea splits as  -RRRRRRRRLR-RRRRRRL-RRRRLRRRRRRRR, agree=0.787, adj=
0.216, (0 split)
##       Distance    < -0.7182662  to the left,  agree=0.782, adj=0.197, (0 split)
##
## Node number 5: 1223 observations,    complexity param=0.03011475
##   mean=0.4343129, MSE=0.7280805
##   left son=10 (730 obs) right son=11 (493 obs)
##   Primary splits:
##       Bathroom    < -0.1740814  to the left,  improve=0.24737120, (0 missing)
##       Bedroom2    < 0.4785597   to the left,  improve=0.19829290, (0 missing)
##       Rooms       < 0.459301    to the left,  improve=0.19121620, (0 missing)
##       Regionname  splits as  RLLLLRL-, improve=0.09174081, (0 missing)
##       CouncilArea splits as  -LLL-L-LLR---LL-L-R-R-LLLLR--LLR-, improve=0.0819578
1, (0 missing)
##   Surrogate splits:
##       Rooms       < 0.459301    to the left,  agree=0.757, adj=0.398, (0 split)
##       Bedroom2    < 0.4785597   to the left,  agree=0.751, adj=0.381, (0 split)
##       Landsize    < -0.1173026  to the left,  agree=0.662, adj=0.162, (0 split)
##       Car         < -0.1544004  to the left,  agree=0.648, adj=0.126, (0 split)
##       CouncilArea splits as  -LLR-L-LLL---RR-L-L-R-LRLLR--LRR-, agree=0.644, adj=
0.118, (0 split)
##
## Node number 6: 781 observations,    complexity param=0.04406257
##   mean=0.2154378, MSE=0.8868169
##   left son=12 (304 obs) right son=13 (477 obs)
##   Primary splits:
##       Type     splits as  RRL, improve=0.4653295, (0 missing)
##       Rooms    < -0.5698475  to the left,  improve=0.4131704, (0 missing)
##       Bedroom2 < -0.5511319  to the left,  improve=0.4106474, (0 missing)
##       Landsize < -0.3158912  to the left,  improve=0.2813159, (0 missing)
##       Bathroom < -0.1740814  to the left,  improve=0.2325725, (0 missing)
##   Surrogate splits:
##       Rooms    < -0.5698475  to the left,  agree=0.846, adj=0.605, (0 split)
##       Bedroom2 < -0.5511319  to the left,  agree=0.834, adj=0.572, (0 split)
```

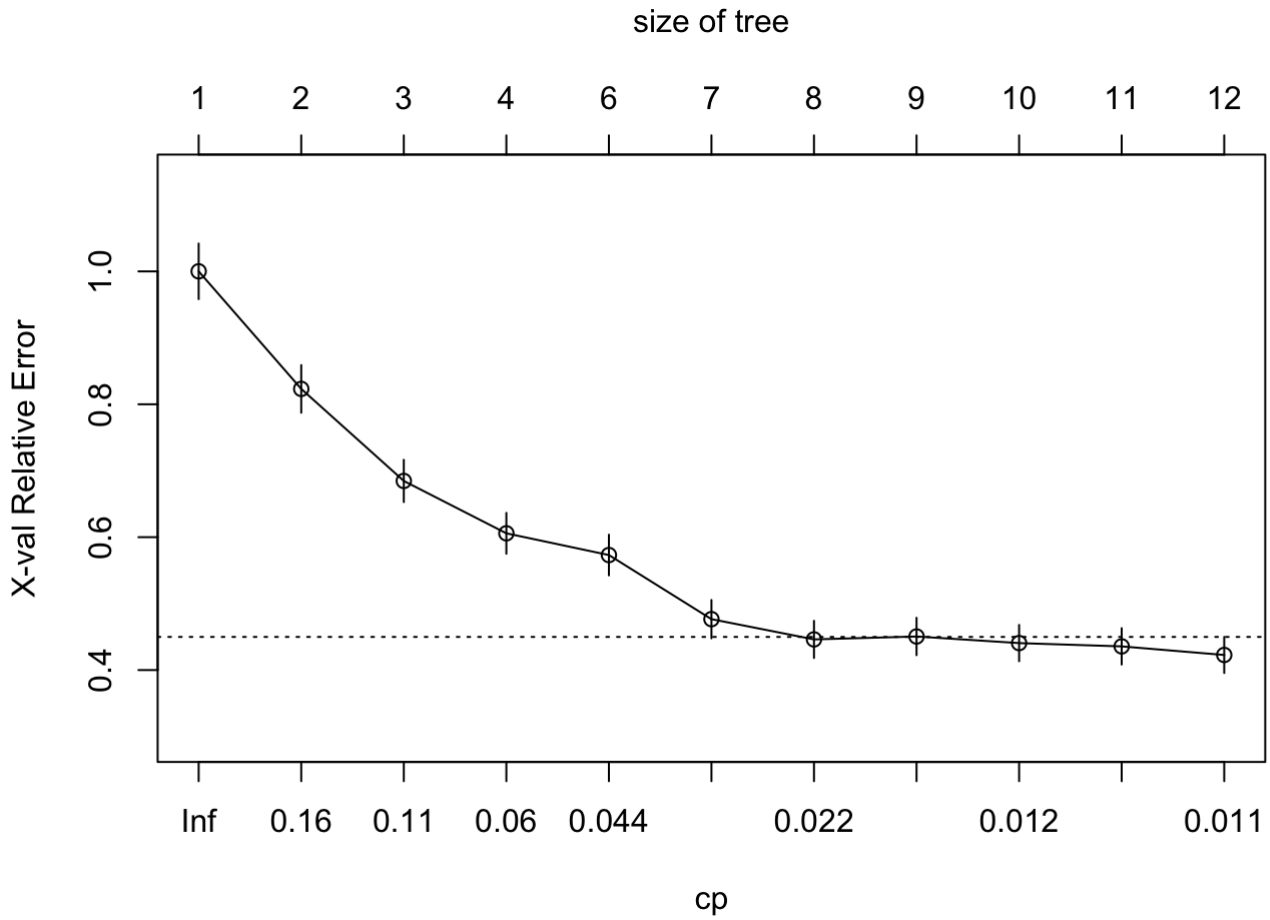
```
##       Landsize < -0.3967977  to the left,  agree=0.805, adj=0.500, (0 split)
##       Car      < -0.1544004  to the left,  agree=0.703, adj=0.237, (0 split)
##       age      < 0.1687262   to the left,  agree=0.681, adj=0.181, (0 split)
##
## Node number 7: 486 observations,    complexity param=0.0166789
##   mean=2.054846, MSE=1.803291
##   left son=14 (467 obs) right son=15 (19 obs)
##   Primary splits:
##       Landsize < 0.6540669   to the left,  improve=0.13920060, (0 missing)
##       Bathroom < 2.634776    to the left,  improve=0.13479430, (0 missing)
##       Distance < -0.6214687  to the right, improve=0.10199870, (0 missing)
##       Rooms    < 1.48845     to the left,  improve=0.07154947, (0 missing)
##       Bedroom2 < 1.508251    to the left,  improve=0.06628877, (0 missing)
##   Surrogate splits:
##       age < 2.19675     to the left,  agree=0.965, adj=0.105, (0 split)
##
## Node number 8: 1340 observations
##   mean=-0.730148, MSE=0.124566
##
## Node number 9: 3601 observations,    complexity param=0.04411059
##   mean=-0.2025513, MSE=0.4184563
##   left son=18 (1908 obs) right son=19 (1693 obs)
##   Primary splits:
##       CouncilArea splits as  -RLRLLRLRRL-LRRLLLR-RLLLRLRLLLLRL, improve=0.2478253
0, (0 missing)
##       Regionname  splits as  RLLLLRLL, improve=0.16764130, (0 missing)
##       Bathroom    < 1.230347    to the left,  improve=0.10307290, (0 missing)
##       Distance    < 0.9421832   to the right, improve=0.08698279, (0 missing)
##       Rooms       < 0.459301    to the left,  improve=0.06696920, (0 missing)
##   Surrogate splits:
##       Regionname    splits as  RLLLRRLL, agree=0.799, adj=0.572, (0 split)
##       Propertycount < -0.1059986  to the right, agree=0.612, adj=0.175, (0 split)
##       Distance      < 0.9421832   to the right, agree=0.609, adj=0.168, (0 split)
##       SellYear      splits as  RLL, agree=0.573, adj=0.092, (0 split)
##       Type          splits as  LRR, agree=0.567, adj=0.079, (0 split)
##
## Node number 10: 730 observations
##   mean=0.08555325, MSE=0.2562275
##
## Node number 11: 493 observations
##   mean=0.950732, MSE=0.9799729
##
## Node number 12: 304 observations
##   mean=-0.5892357, MSE=0.2055655
##
## Node number 13: 477 observations
##   mean=0.7282695, MSE=0.6453312
##
## Node number 14: 467 observations,    complexity param=0.01157281
##   mean=1.953788, MSE=1.418461
##   left son=28 (391 obs) right son=29 (76 obs)
##   Primary splits:
##       Distance      < -0.6214687  to the right, improve=0.12778510, (0 missing)
##       Bathroom      < 2.634776    to the left,  improve=0.11507930, (0 missing)
##       Landsize      < 0.2118395   to the left,  improve=0.09645833, (0 missing)
##       Propertycount < 0.584189    to the left,  improve=0.06706083, (0 missing)
##       age           < 1.236819    to the left,  improve=0.05252426, (0 missing)
##   Surrogate splits:
```

```
##       Propertycount < 0.7904309   to the left,  agree=0.878, adj=0.250, (0 split)
##       age           < 1.899307    to the left,  agree=0.848, adj=0.066, (0 split)
##       Type          splits as  LLR, agree=0.839, adj=0.013, (0 split)
##
## Node number 15: 19 observations
##   mean=4.538752, MSE=4.841212
##
## Node number 18: 1908 observations,    complexity param=0.01115889
##   mean=-0.5058965, MSE=0.1522951
##   left son=36 (1558 obs) right son=37 (350 obs)
##   Primary splits:
##       Distance    < -0.2938464  to the right, improve=0.28088750, (0 missing)
##       CouncilArea splits as  --L-RR-R--L-L--LRL---LLL-L-LLRL-L, improve=0.2224488
0, (0 missing)
##       Bathroom    < 1.230347    to the left,  improve=0.07257149, (0 missing)
##       age         < 0.3174479   to the left,  improve=0.06235106, (0 missing)
##       Bedroom2    < 0.4785597   to the left,  improve=0.05634074, (0 missing)
##   Surrogate splits:
##       CouncilArea splits as  --L-LL-R--L-L--LLL---LLL-L-LLRL-L, agree=0.887, adj=
0.383, (0 split)
##       Landsize    < -0.3535863  to the right, agree=0.833, adj=0.089, (0 split)
##       age         < 0.4391294   to the left,  agree=0.829, adj=0.069, (0 split)
##       Bedroom2    < -0.5511319  to the right, agree=0.818, adj=0.009, (0 split)
##
## Node number 19: 1693 observations,    complexity param=0.0115827
##   mean=0.1393168, MSE=0.4978404
##   left son=38 (1430 obs) right son=39 (263 obs)
##   Primary splits:
##       Bathroom    < 1.230347    to the left,  improve=0.10051670, (0 missing)
##       Rooms       < 0.459301    to the left,  improve=0.09533022, (0 missing)
##       Bedroom2    < 0.4785597   to the left,  improve=0.08534718, (0 missing)
##       CouncilArea splits as  -L-L--R-LR---RL---R-R---R-L----R-, improve=0.0724742
8, (0 missing)
##       Landsize    < 0.07944715  to the left,  improve=0.06104191, (0 missing)
##   Surrogate splits:
##       Rooms    < 1.48845     to the left,  agree=0.874, adj=0.186, (0 split)
##       Bedroom2 < 1.508251    to the left,  agree=0.871, adj=0.171, (0 split)
##       age      < -1.359052   to the right, agree=0.848, adj=0.019, (0 split)
##
## Node number 28: 391 observations
##   mean=1.766087, MSE=1.00555
##
## Node number 29: 76 observations
##   mean=2.919462, MSE=2.428995
##
## Node number 36: 1558 observations
##   mean=-0.6039266, MSE=0.07201731
##
## Node number 37: 350 observations
##   mean=-0.0695225, MSE=0.2764459
##
## Node number 38: 1430 observations
##   mean=0.04338246, MSE=0.3975464
##
## Node number 39: 263 observations
##   mean=0.6609369, MSE=0.7210361
```

Results

From above model summary report, we can see that 11 variables are used to construct the tree with 11 internal nodes resulting in 12 terminal nodes and MSE on the test data is 0.4012.

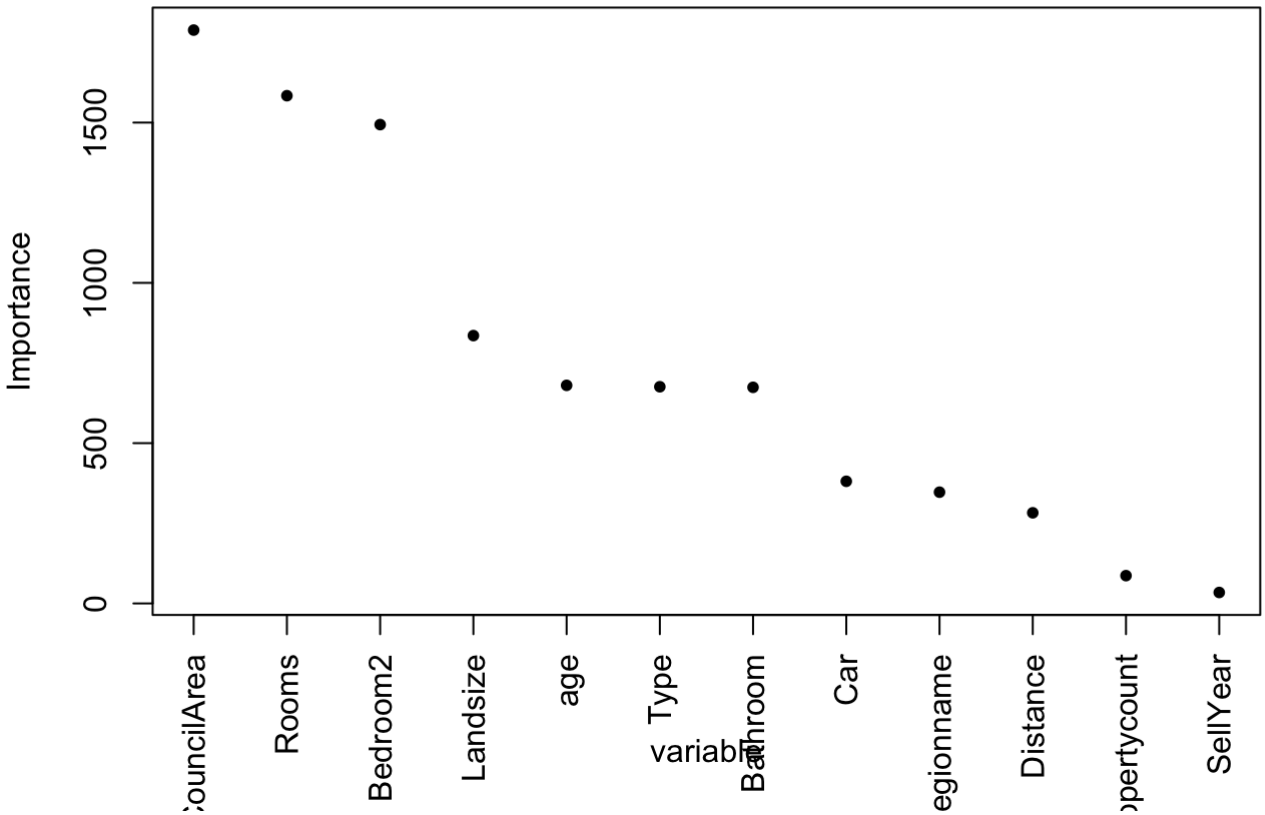
```
plotcp(RT_fit)
```



Interpretation

- Now we find diminishing returns after 12 terminal nodes as illustrated:
  - y-axis is cross validation error
  - lower x-axis is cost complexity (α) value
  - upper x-axis is the number of terminal nodes (tree size = |T|)
  - The dashed line which goes through the point |T|=9.
- Rpart is performing some automated tuning, with an optimal subtree of 11 splits, 12 terminal nodes, and a cross-validated error of 0.4227.
- Thus, we could use a tree with 9 terminal nodes.

```
argPlot <- data.frame(RT_fit$variable.importance)

plot(RT_fit$variable.importance, xlab="variable",
    ylab="Importance", xaxt = "n", pch=20)
axis(1, at=1:12, labels=row.names(argPlot), las=2)
```



setting a 10 fold cross- validation

```
RT_cv <- rpart(
    formula = Price ~ .,
    data    = train,
    method  = "anova",
    control = list(xval = 10)
)
```

```
predictions3 <- predict(RT_cv,train)
Rmes2 <- RMSE(predictions3, train$Price)
MSE2_train=Rmes2^2
predictions4 <- predict(RT_cv, test)
Rmes3 <- RMSE(predictions4, test$Price)
MSE2_test=Rmes3^2
MSE2_train
```

```
## [1] 0.3827541
```

```
MSE2_test
```

```
## [1] 0.4011495
```

Result

After a 10 fold cross-validation,Train MSE:0.3827 ; Test MSE: 0.4011.

Bagging

train bagged model

```
# make bootstrapping reproducible
set.seed(820)

bagged_m1 <- bagging(
  formula = Price ~ .,
  data    = train,
  coob    = TRUE
)

bagged_m1
```

```
##
## Bagging regression trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = Price ~ ., data = train, coob = TRUE)
##
## Out-of-bag estimate of root mean squared error:  0.5964
```

```
pred <- predict(bagged_m1, newdata = test)
Rmse_bagging <- rmse(pred = pred, test$Price)
MSE3_test_bagging=Rmse_bagging^2
MSE3_test_bagging
```

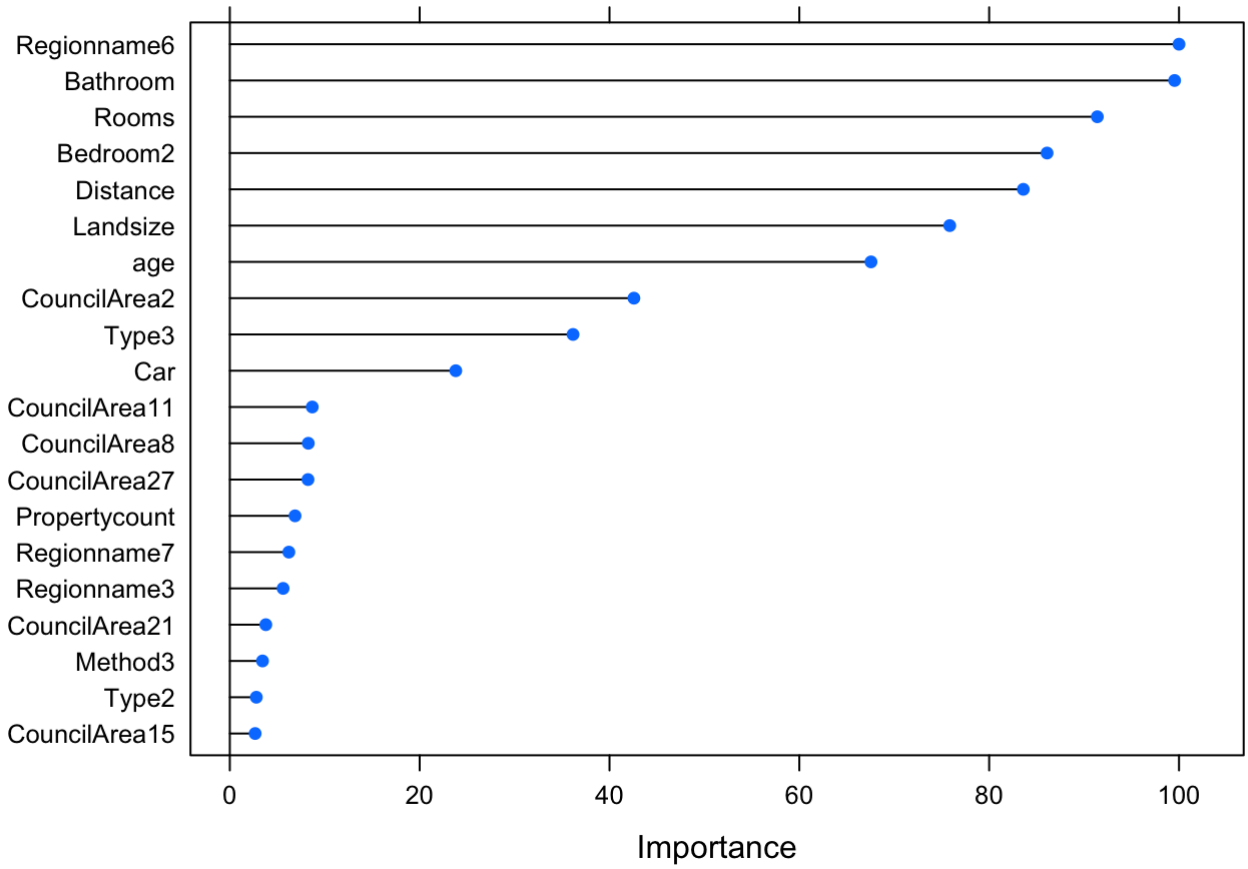
```
## [1] 0.3442464
```

CV bagged model

```
# Specify 10-fold cross validation
ctrl <- trainControl(method = "cv",  number = 10)


bagged_cv <- train(
  Price ~ .,
  data = train,
  method = "treebag",
  trControl = ctrl,
  importance = TRUE
  )

# assess results
bagged_cv
```

```
## Bagged CART
##
## 7431 samples
##   14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6688, 6687, 6689, 6686, 6687, 6689, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.6149614  0.6199037  0.4100107
```

```
plot(varImp(bagged_cv), 20)
```




```
pred4 <- predict(bagged_cv, test)
Rmse4 <- rmse(pred4, test$Price)
MSE4_test_bagging <- Rmse4^2
MSE4_test_bagging
```

```
## [1] 0.3956213
```

Random Forests

ntree=500 (default)

```
set.seed(810)

rf <-
  randomForest(formula = Price ~ .,
               data = train,
               importance = TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Price ~ ., data = train, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.2190135
##                     % Var explained: 77.75
```

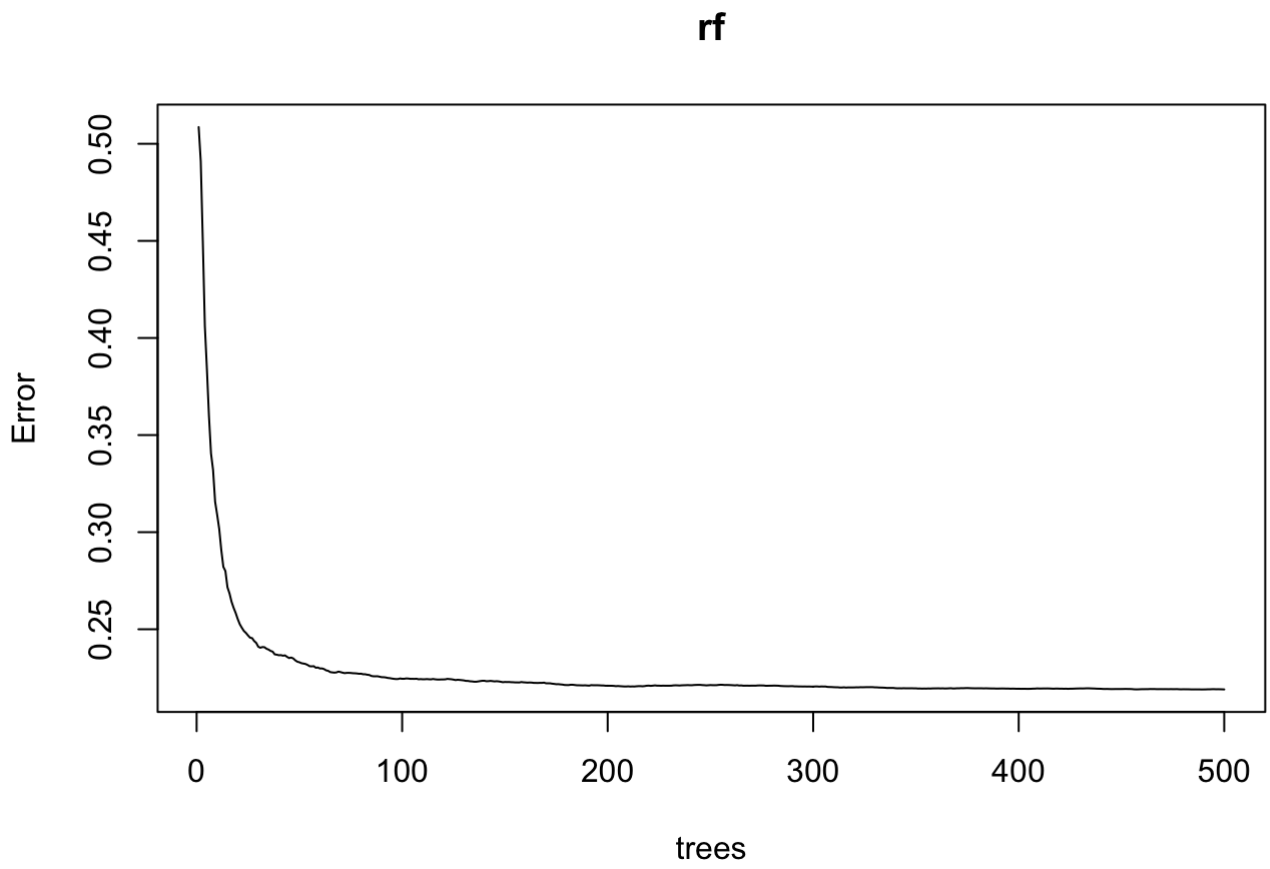
The above MSE and Variance explained are calculated using Out-of-Bag(OOB) error estimation. The number of variables randomly selected at each split is 4 (default).

```
# compute test MSE
yhat.rf <- predict(rf, newdata = test)
mse.rf <- mean((yhat.rf - test$Price) ^ 2)
mse.rf
```

```
## [1] 0.2126807
```

The test set MSE is 0.2126807; this indicates that random forests yielded an improvement over bagging.

```
# plot the error vs number of trees
plot(rf)
```



Interpretation

From the above figure, we can notice how the error decreases as we add more trees. We found that the model's average error starts to stabilize at about 100 trees, and the average error decreases at a slower rate, starting at about 300 trees.

Find the number of trees that minimize the MSE

```
# rf0$mse
which.min(rf$mse)
```

```
## [1] 489
```

The RMSE of the optimal random forest is (the error of the average Price)

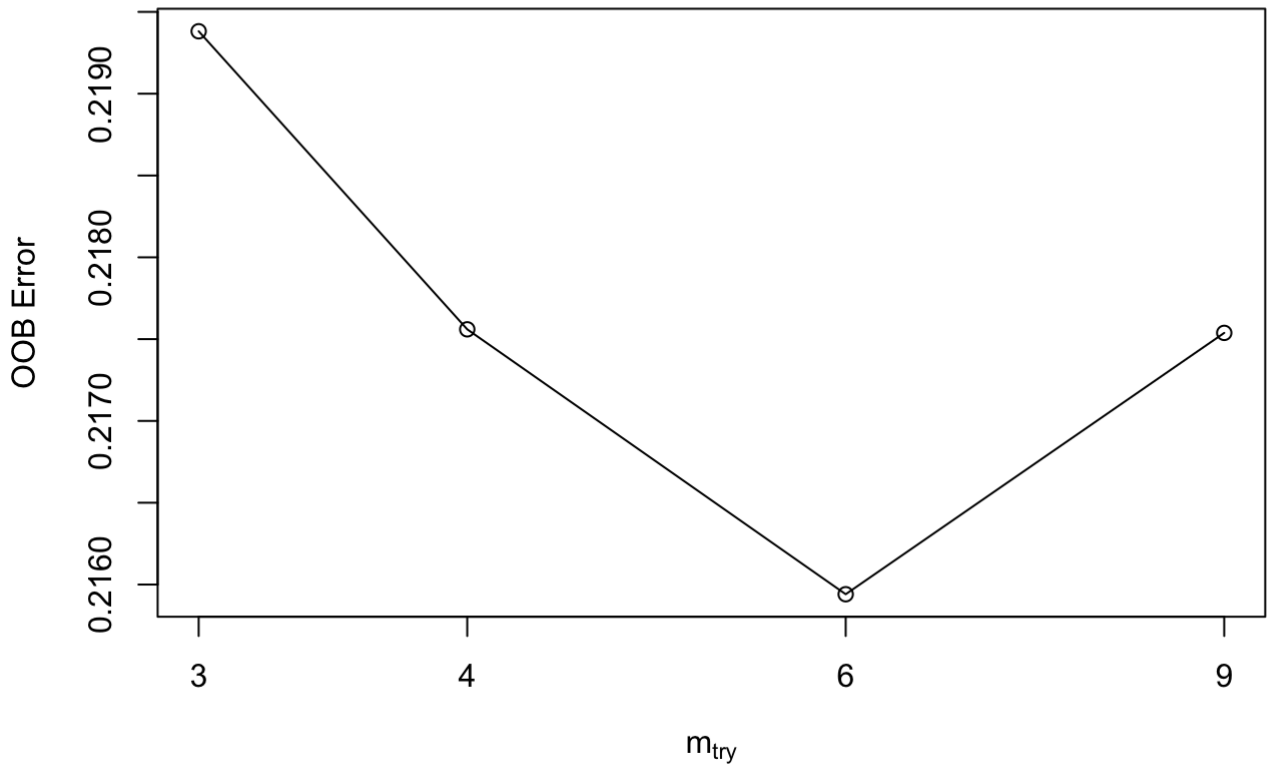
```
mse.rf.opt <- rf$mse[which.min(rf$mse)]
mse.rf.opt
```

```
## [1] 0.2189928
```

Tune RandomForest For The Optimal mtry Parameter

```
set.seed(810)

# plot the OOB error value corresponding to each mtry value
mtry_opt <- tuneRF(
  x = train[,-c('Price')],
  y = train$Price,
  stepFactor = 1.5,
  improve = 1e-5,
  ntree = 500
)
```

```
## mtry = 4  OOB error = 0.2175599
## Searching left ...
## mtry = 3     OOB error = 0.2193815
## -0.008372863 1e-05
## Searching right ...
## mtry = 6     OOB error = 0.2159408
## 0.007441885 1e-05
## mtry = 9     OOB error = 0.2175382
## -0.007397135 1e-05
```




```
mtry_opt
```

```
##   mtry  OOBError
## 3    3 0.2193815
## 4    4 0.2175599
## 6    6 0.2159408
## 9    9 0.2175382
```

The most accurate value for mtry was six, with the lowest OOB Error of 0.2159408. Thus we choose six as the optimal mtry level.

Set ntree=500, mtry=6

```
set.seed(810)

rf2 <-
  randomForest(
    formula = Price ~ .,
    data = train,
    mtry = 6,
    importance = TRUE
  )
rf2
```

```
##
## Call:
##  randomForest(formula = Price ~ ., data = train, mtry = 6, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 0.2160597
##                     % Var explained: 78.05
```

```
# compute test MSE
yhat.rf2 <- predict(rf2, newdata = test)
mse.rf2 <- mean((yhat.rf2 - test$Price) ^ 2)
mse.rf2
```

```
## [1] 0.2115062
```

```
# compare rf (mtry=4) v.s. rf2 (mtry=6)
print(mse.rf) # mtry=4 (default)
```

```
## [1] 0.2126807
```

```
print(mse.rf2) # mtry=6
```

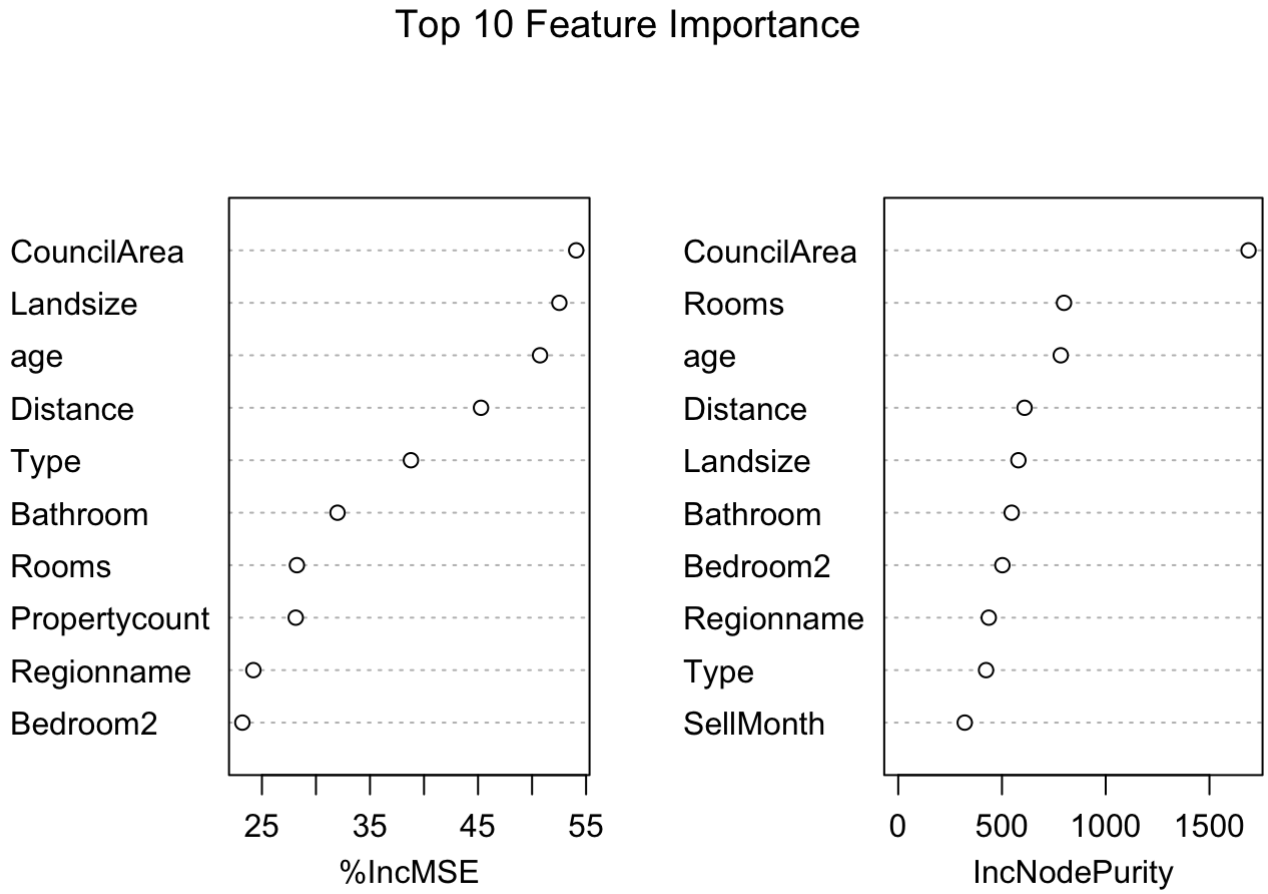
```
## [1] 0.2115062
```

The test set MSE is 0.2115062; this indicates that random forests yielded an improvement when `mtry=6` over `mtry=4` in this case.

```
# View the importance of each variable
importance(rf2)
```

```
##                   %IncMSE IncNodePurity
## Rooms          28.236506717     798.97720
## Type           38.787836662     423.32872
## Method          4.032808757     100.24231
## Distance       45.273557453     608.95794
## Bedroom2       23.188694391     501.99143
## Bathroom       31.994683968     546.87787
## Car            17.492084776     110.68837
## Landsize       52.524638732     579.17912
## CouncilArea    54.084471377    1688.40411
## Regionname     24.208334041     436.28853
## Propertycount  28.126708567     194.95971
## SellYear        6.586972312      47.35067
## SellMonth      -0.002773935     320.56163
## age            50.737487069     783.28740
```

```
# plot top 10 importance measures
varImpPlot(rf2,
           sort = T,
           n.var = 10,
           main = 'Top 10 Feature Importance')
```



Interpretation

The results indicate that across all of the trees considered in the random forest, the `CouncilArea` is the most important variables.

Boosting

(1) Boosting 1 (default `n.trees` and `shrinkage` parameters)

```
set.seed(810)
boost1 <-
  gbm(
    Price ~ .,
    data = train,
    distribution = "gaussian",
    interaction.depth = 2,
    cv.folds = 10
  )

# compute test MSE
yhat.boost1 <- predict(boost1, newdata = test, n.trees = 100)
mse.boost1 <- mean((yhat.boost1 - test$Price) ^ 2)
mse.boost1
```

```
## [1] 0.252532
```

(2) Boosting 2.1 (set `n.trees=500`)

```
set.seed(810)
boost2 <-
  gbm(
    Price ~ .,
    data = train,
    distribution = "gaussian",
    n.trees = 500,
    interaction.depth = 2,
    cv.folds = 10
  )

# compute test MSE
yhat.boost2 <- predict(boost2, newdata = test, n.trees = 500)
mse.boost2 <- mean((yhat.boost2 - test$Price) ^ 2)
mse.boost2
```

```
## [1] 0.2072101
```

The test MSE obtained is 0.2073137, which is higher than the first model using 100 trees.

(3) Boosting 2.2 (set `n.trees=1000`)

```
set.seed(810)
boost3 <-
  gbm(
    Price ~ .,
    data = train,
    distribution = "gaussian",
    n.trees = 1000,
    interaction.depth = 2,
    cv.folds = 10
  )

# compute test MSE
yhat.boost3 <- predict(boost3, newdata = test, n.trees = 1000)
mse.boost3 <- mean((yhat.boost3 - test$Price) ^ 2)
mse.boost3
```

```
## [1] 0.2002709
```

The test MSE obtained is 0.1992171. In this case, using `n.trees = 1000` leads to a slightly lower test MSE than `n.trees = 500`.

(4) Boosting 2.3 (set `n.trees=5000`)

```
set.seed(810)
boost4 <-
  gbm(
    Price ~ .,
    data = train,
    distribution = "gaussian",
    n.trees = 5000,
    interaction.depth = 2,
    cv.folds = 10
  )

# compute test MSE
yhat.boost4 <- predict(boost4, newdata = test, n.trees = 5000)
mse.boost4 <- mean((yhat.boost4 - test$Price) ^ 2)
mse.boost4
```

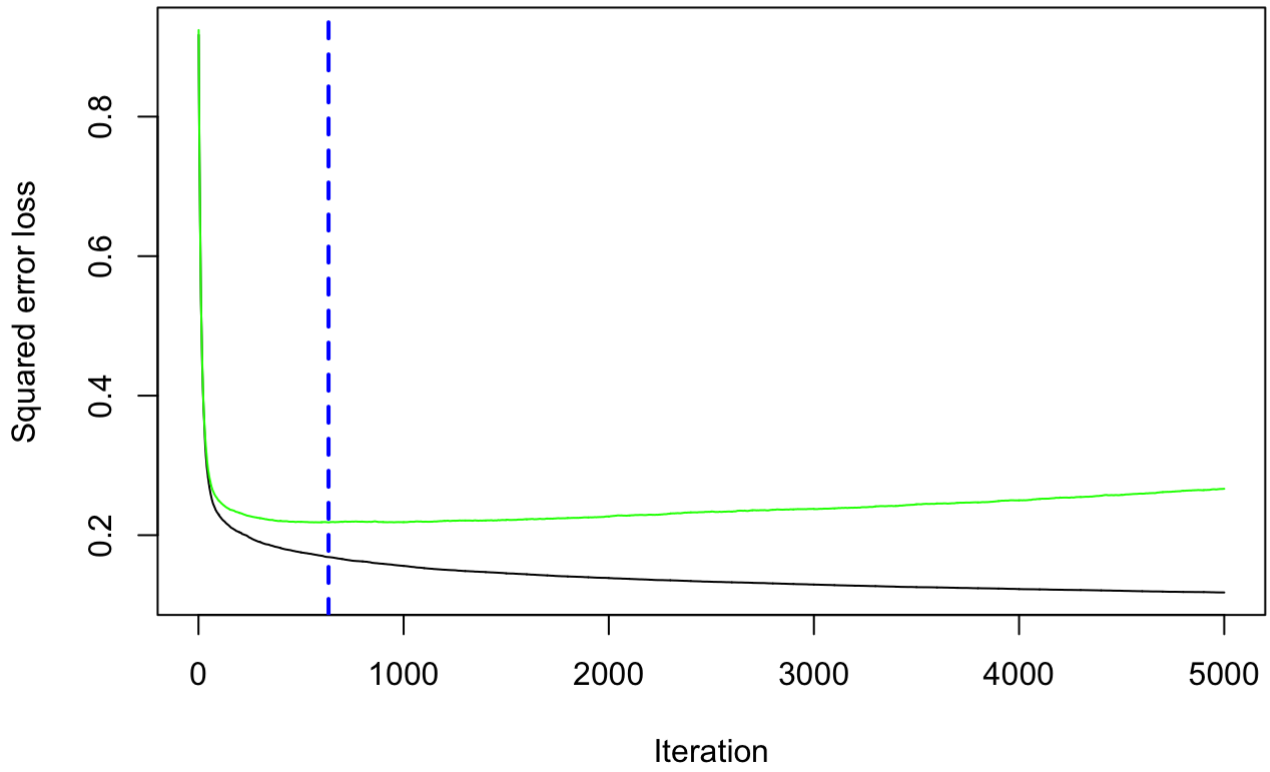
```
## [1] 0.2013821
```

The test MSE obtained is 0.2033423. Increasing the number of trees (iteration) doesn't lower test MSE, thus we will modify other parameters using 1000 trees.

```
# find index for n trees with minimum CV error
mse_err <- which.min(boost4$cv.error)
mse_err
```

```
## [1] 634
```

```
# plot loss function as a result of n trees added to the ensemble
gbm.perf(boost4, method = "cv")
```




```
## [1] 634
```

```
# compute MSE
mse_min <- boost4$cv.error[mse_err]
mse_min
```

```
## [1] 0.2182741
```

We would like to look for the optimal iteration with minimum cross-validation error when we run the model with 5000 trees, and here we plot a loss function as a result of n trees added to the model. When `n.trees` was 634, the model has the minimum error.

(5) Boosting 3.1 (set `shrinkage=0.2`)

```
set.seed(810)
boost5 <-
  gbm(
    Price ~ .,
    data = train,
    distribution = "gaussian",
    n.trees = 1000,
    interaction.depth = 2,
    shrinkage = 0.2,
    cv.folds = 10
  )

# MSE
yhat.boost5 <- predict(boost5, newdata = test, n.trees = 1000)
mse.boost5 <- mean((yhat.boost5 - test$Price) ^ 2)
mse.boost5
```

```
## [1] 0.2060213
```

The test MSE obtained is 0.203148. In this case, using `shrinkage = 0.2` doesn't improve test MSE than `shrinkage = 0.1`.

```
# compare all the parameter settings
print(mse.boost1)
```

```
## [1] 0.252532
```

```
print(mse.boost2)
```

```
## [1] 0.2072101
```

```
print(mse.boost3)
```

```
## [1] 0.2002709
```

```
print(mse.boost4)
```

```
## [1] 0.2013821
```

```
print(mse.boost5)
```

```
## [1] 0.2060213
```

Result

Here we will choose mse.boost3 (sets `n.trees=1000` and `shrinkage=0.1`) when we tune manually, which has the lowest test MSE among five models. It is also superior than the test MSE for bagging and random forest.

Grid Search

```
# define hyperparameter grid
hyperparams <- expand.grid(n.trees = c(500, 1000),
                           interaction.depth = c(2, 3),
                           shrinkage = c(0.1, 0.2),
                           n.minobsinnode = 10)

# apply hyperparameter grid to train()
set.seed(810)
gbm.gridsearch <- train(Price ~ .,
                   data = train,
                   method = "gbm",
                   trControl = trainControl(method = "repeatedcv", number = 10, repea
ts = 3),
                   verbose = FALSE,
                   tuneGrid = hyperparams)
```

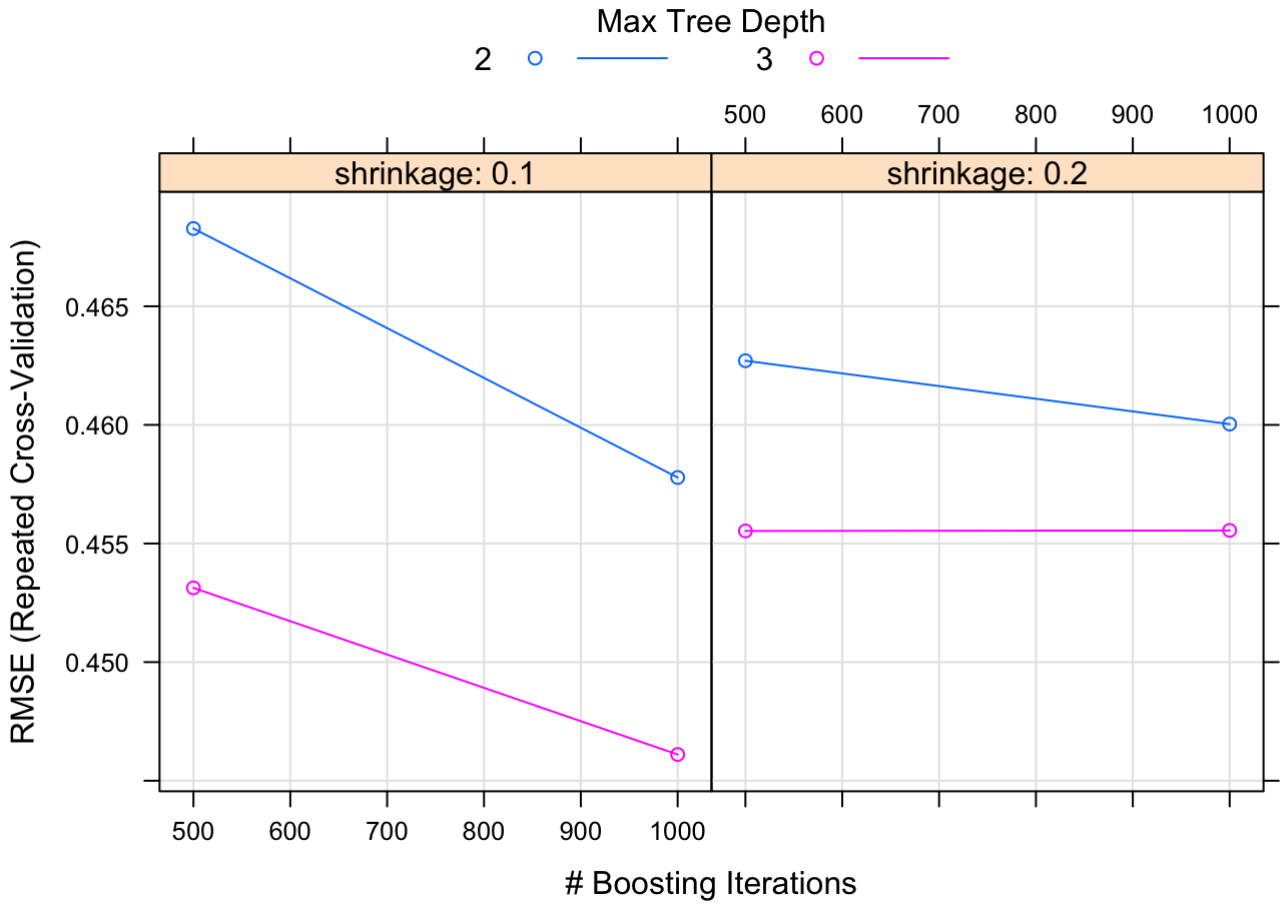
Explanation

We manually define hyperparameters as a grid and apply hyperparameter grid to `train()` function. We define the following hyperparameter grid for a Gradient Boosting Model: the number of trees as 500 and 1000; the tree complexity as 2 and 3; the learning rate as 0.1 and 0.2 and the minimum number of training set samples in a node to commence splitting as 10.

```
print(gbm.gridsearch)
```

```
## Stochastic Gradient Boosting
##
## 7431 samples
##   14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 6689, 6688, 6689, 6688, 6687, 6688, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  RMSE       Rsquared   MAE
##   0.1        2                   500     0.4682779  0.7767708  0.2828241
##   0.1        2                  1000     0.4577827  0.7863398  0.2757786
##   0.1        3                   500     0.4531315  0.7905932  0.2721552
##   0.1        3                  1000     0.4461076  0.7969450  0.2662210
##   0.2        2                   500     0.4627059  0.7817740  0.2807868
##   0.2        2                  1000     0.4600345  0.7845807  0.2795260
##   0.2        3                   500     0.4555301  0.7885150  0.2740519
##   0.2        3                  1000     0.4555548  0.7890756  0.2734704
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 1000, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
plot(gbm.gridsearch)
```




```
mse.gbm.gridsearch <- 0.4461076^2
mse.gbm.gridsearch
```

```
## [1] 0.199012
```

Interpretation

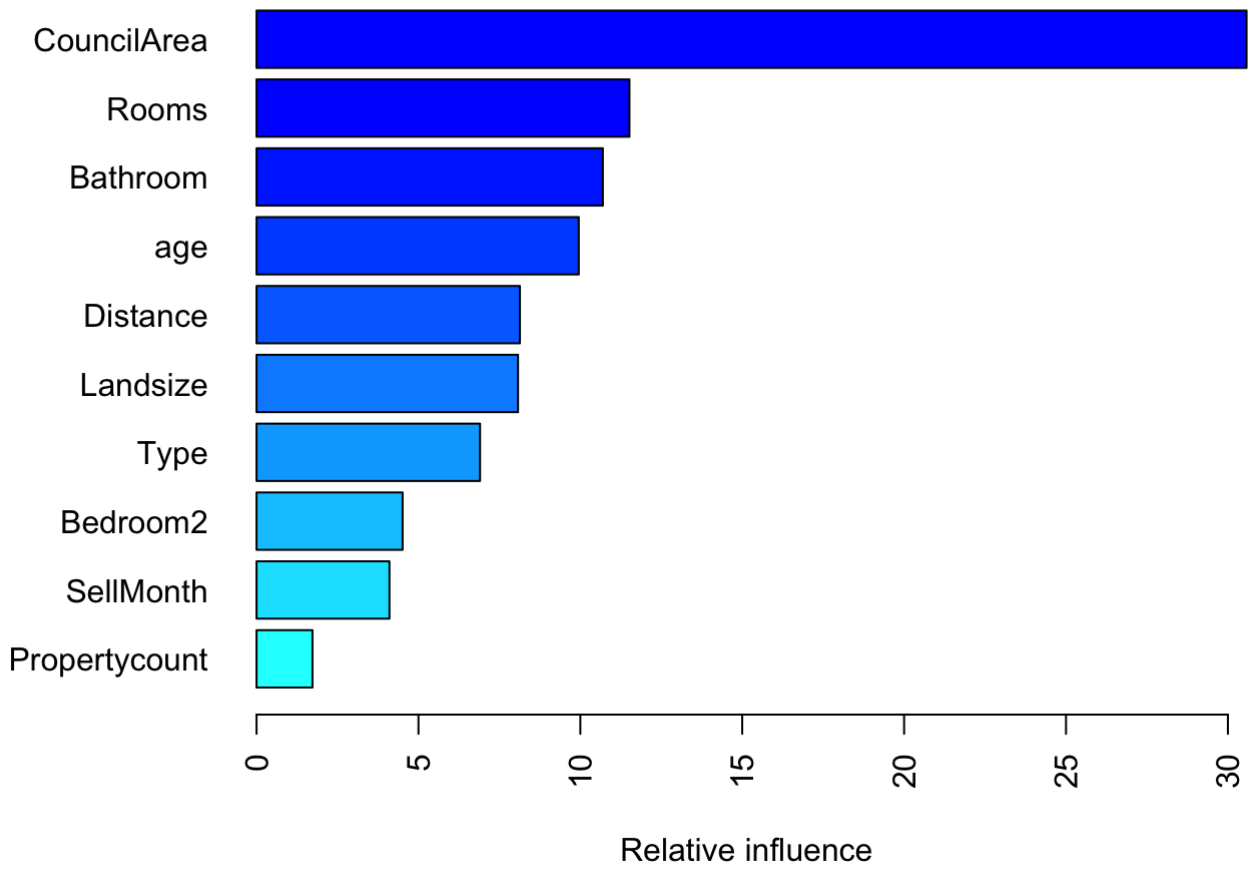
Each axis of the grid is a parameter, and each row in the table is a specific combination of parameters. We have eight combinations with tuning three parameters. We can see that the final values used for the model were `n.trees = 1000`, `interaction.depth = 3`, `shrinkage = 0.1` and `n.minobsinnode = 10`. This optimal model generates the lowest RMSE, 0.4461076.

Examine feature importance

```
set.seed(810)
boost6 <-
  gbm(
    Price ~ .,
    data = train,
    distribution = "gaussian",
    n.trees = 1000,
    interaction.depth = 3,
    shrinkage = 0.1,
    cv.folds = 10
  )

# compute test MSE
yhat.boost6 <- predict(boost6, newdata = test, n.trees = 1000)
mse.boost6 <- mean((yhat.boost6 - test$Price) ^ 2)
mse.boost6
```

```
## [1] 0.194209
```

```
par(mar = c(5, 8, 1, 1))
summary(
  boost6,
  cBars = 10,
  method = relative.influence,
  las = 2
  )
```



```
##                          var    rel.inf
## CouncilArea     CouncilArea 30.5757747
## Rooms                 Rooms 11.5169859
## Bathroom           Bathroom 10.6958440
## age                     age  9.9517031
## Distance           Distance  8.1329672
## Landsize           Landsize  8.0750785
## Type                   Type  6.9021181
## Bedroom2           Bedroom2  4.5110026
## SellMonth         SellMonth  4.1045010
## Propertycount Propertycount  1.7268418
## Car                     Car  1.4785744
## Regionname       Regionname  1.1898609
## Method              Method  0.8963123
## SellYear           SellYear  0.2424355
```

Interpretation

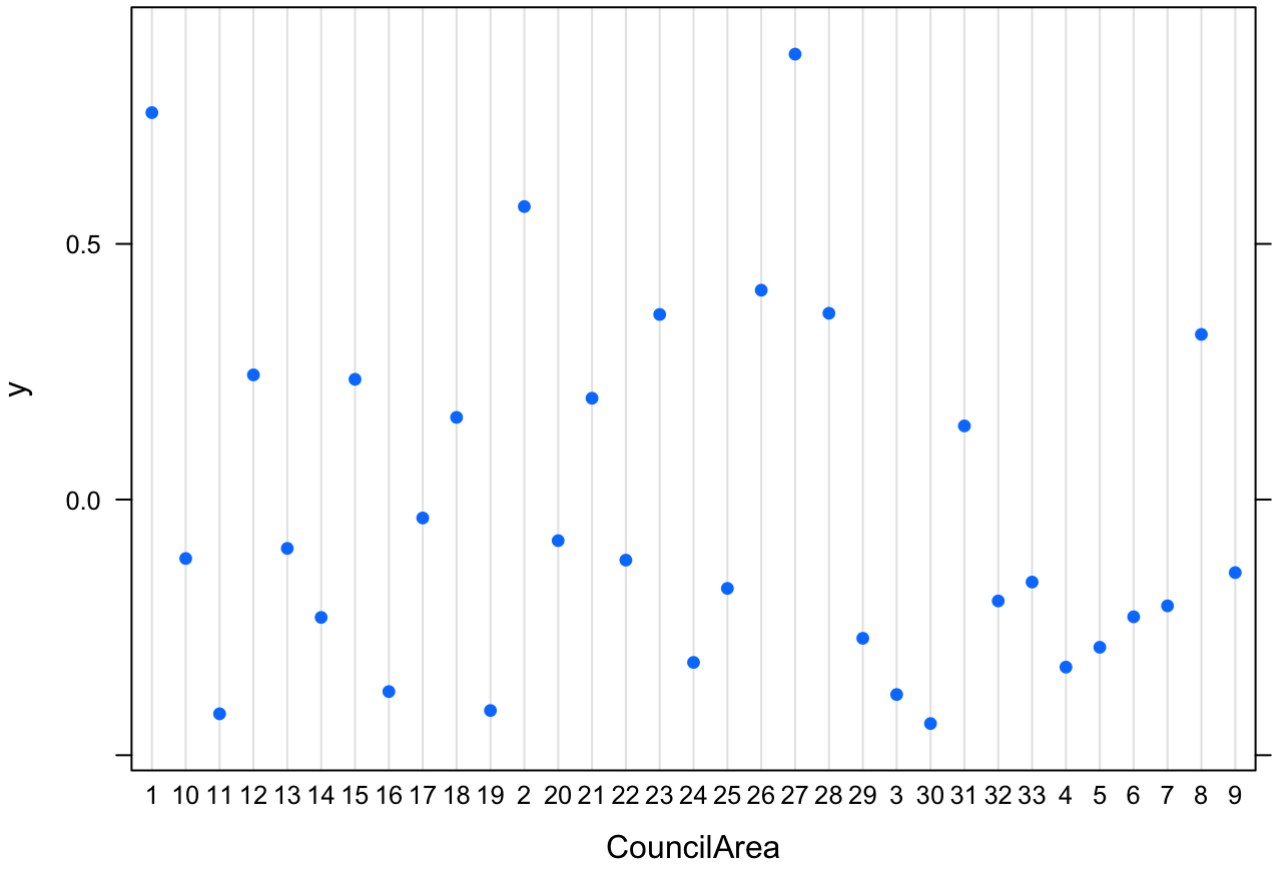
The variable `CouncilArea` has the highest relative influence, indicating that `CouncilArea` is the most important variable.

```
relative.influence(boost6)
```

```
## n.trees not given. Using 633 trees.
```

```
##         Rooms          Type        Method      Distance      Bedroom2
##    2526.00363    1518.03210     151.75604    1688.31560     978.68479
##      Bathroom           Car      Landsize   CouncilArea    Regionname
##    2273.46663     289.76457    1519.59849    6356.00670     210.40075
## Propertycount      SellYear     SellMonth           age
##     289.11309      49.56561     649.47773    2095.09355
```

```
# par(mfrow = c(1, 3))
plot(boost6, i = "CouncilArea")
```

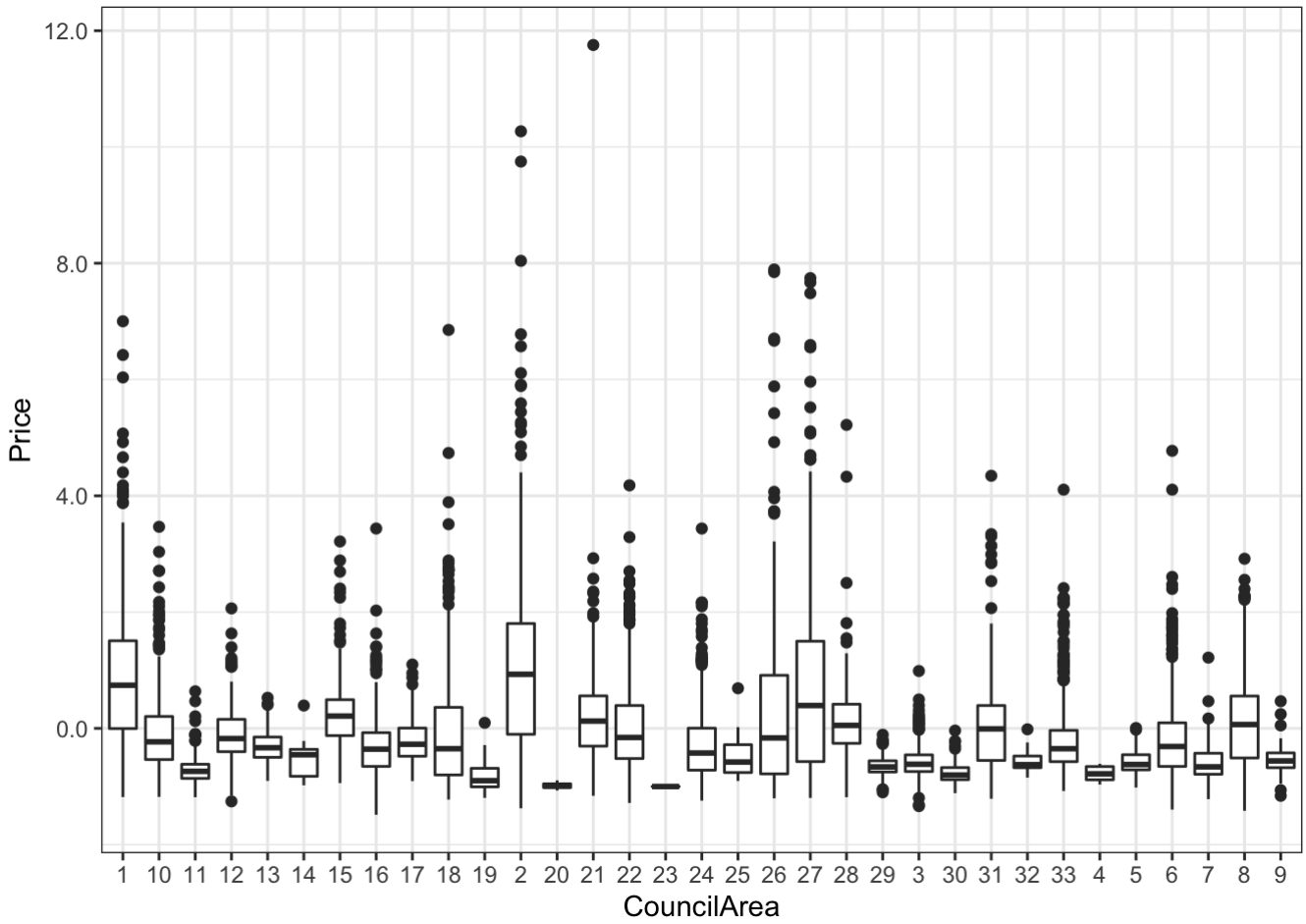


Interpretation

We examine the effect of the `CouncilArea` feature on the predicted prices. According to the graph, we find that `CouncilArea=27` (Stonnington City Council) predicts the highest house prices, which is followed by `CouncilArea=1` (Bayside City Council) and `CouncilArea=2` (Boroondara City Council).

Compare to the distribution of CouncilArea over Price:

```
ggplot(data=dd, aes(x=factor(CouncilArea), y=Price))+
        geom_boxplot() + labs(x='CouncilArea') +
        scale_y_continuous(labels = comma)
```



Interpretation

By comparing the above two graphs, `CouncilArea` within no.1, 2, 27 show the top 3 effects on house prices. This boxplot provides a rough reference to the output of plot of boost6, which demonstrates the effect of the `CouncilArea` variable on the predicted prices.

Conclusion

```
models <- c('Ridge Regression', 'Lasso Regression', 'Linear Regression', 'Regression
 Trees', 'Bagging', 'Randon Forests', 'Boosting')
test_MSE <- c(mse.test.ridge, mse.test.lasso, mse.test.m1, MSE1_test, MSE4_test_baggi
ng, mse.rf2, mse.boost6)
summary <- data.table(models, test_MSE)
summary
```

```
##               models  test_MSE
## 1:  Ridge Regression 0.3671983
## 2:  Lasso Regression 0.3563141
## 3: Linear Regression 0.3560723
## 4:  Regression Trees 0.4011495
## 5:           Bagging 0.3956213
## 6:    Randon Forests 0.2115062
## 7:          Boosting 0.1942090
```

- According to the exploratory data analysis, we learned that some patterns or relationships exist among the location's features and house prices, such as `Regionname` ; and the houses far from CBD tend to have lower prices. After we execute several models, `CouncilArea` does strongly affect the house prices.

- After comparing all the models, the results indicate that `CouncilArea` has a stronger effect on Melbourne's housing price than other features.
- Comparing all the models with test MSE, regression tree has the highest test MSE, followed by Ridge, Lasso, linear regression, and bagging. Comparatively, random forests and boosting have much lower test MSE.
- If we compare the last two models, random forests and boosting, they excel in different areas. Random forest tends to perform better with data with a lot of statistical noise. While boosting tends to perform with unbalanced data, which was reduced due to our scaling.