Data Wrangling, randomForest, GradientBoosting

Marc Kullmann

2019-08-30

Contents

1 Introduction **2**

2 Data Overview and Cleaning **2**

2.1 Import data, declare tibble, overview . . . 2

2.2 Missing Data . . . 3

2.3 Data Cleaning . . . 4

2.4 Categorical Variable Adjustment . . . 5

2.4.1 Missing variables treatment . . . 10

3 Exploratory Data Analysis **15**

3.1 Summary . . . 15

3.2 Frequency Plots . . . 17

3.3 Plot average daily Price . . . 19

3.4 Location Plots . . . 20

3.5 Price Evolution for each District by Year . . . 22

3.6 Analysis . . . 24

3.6.1 TTest . . . 24

3.6.2 Linear Regression . . . 25

3.6.3 RandomForest Analysis . . . 27

3.6.4 Gradient Boosting Time Series Analysis . . . 28

4 References **31**

1 Introduction

This markdown is a small attempt to get used to the evironment of R and the machine learning algorithm of random forest, hereby I used the dataset from kaggle (https://www.kaggle.com/ruiqurm/lianjia). As one can find, several enthusiasts have conducted comprehensive analysis on this dataset, however, I have worked mostly on my own.

2 Data Overview and Cleaning

2.1 Import data, declare tibble, overview

First of all, we have to import the data file. For convenience, I choose to declare the CSV file as a tibble dataframe.

```
Df <- read.csv("new.csv",
               sep=',', fileEncoding="latin1")
data <- as_tibble(Df)
```

Get a quick overview of the underlaying data set.

```
str(data)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    318851 obs. of  26 variables:
##  $ url                : Factor w/ 318851 levels "https://bj.lianjia.com/chengjiao/10108478203
##  $ id                 : Factor w/ 318851 levels "101084782030",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Lng                : num  116 116 117 116 116 ...
##  $ Lat                : num  40 39.9 39.9 40.1 39.9 ...
##  $ Cid                : num  1.11e+12 1.11e+12 1.11e+12 1.11e+12 1.11e+12 ...
##  $ tradeTime          : Factor w/ 2560 levels "2002-06-01","2002-07-06",..: 2046 2034 2170 2
##  $ DOM                : num  1464 903 1271 965 927 ...
##  $ followers          : int  106 126 48 138 286 57 167 138 218 134 ...
##  $ totalPrice         : num  415 575 1030 298 392 ...
##  $ price              : int  31680 43436 52021 22202 48396 52000 37672 49521 27917 55883 ..
##  $ square             : num  131 132 198 134 81 ...
##  $ livingRoom         : Factor w/ 11 levels "#NAME?","0","1",..: 4 4 5 5 4 3 4 5 3 3 ...
##  $ drawingRoom        : Factor w/ 22 levels "¸ß 12","¸ß 14",..: 12 13 13 12 12 11 12 13 11 1
##  $ kitchen            : int  1 1 1 1 1 1 1 1 1 0 ...
##  $ bathRoom           : Factor w/ 18 levels "0","1","1990",..: 2 6 13 2 2 2 2 6 2 1 ...
##  $ floor              : Factor w/ 203 levels "¸Ö»ì½á¹¹","¸ß 10",..: 18 14 196 48 200 202 30
##  $ buildingType       : num  1 1 4 1 4 4 4 1 3 1 ...
##  $ constructionTime   : Factor w/ 74 levels "0","1","1906",..: 62 61 62 65 17 62 54 61 66 66
##  $ renovationCondition: int  3 4 3 1 2 3 4 4 1 4 ...
##  $ buildingStructure  : int  6 6 6 6 2 6 2 6 2 6 ...
##  $ ladderRatio        : num  0.217 0.667 0.5 0.273 0.333 0.333 0.5 0.667 0.333 0.308 ...
##  $ elevator           : num  1 1 1 1 0 1 0 1 0 1 ...
##  $ fiveYearsProperty  : num  0 1 0 0 1 1 0 1 0 1 ...
```

```
##  $ subway              : num  1 0 0 0 1 0 0 0 0 1 ...
##  $ district            : int  7 7 7 6 1 7 7 7 13 1 ...
##  $ communityAverage    : num  56021 71539 48160 51238 62588 ...
```

```r
head(data)
```

```
## # A tibble: 6 x 26
##   url   id      Lng   Lat     Cid tradeTime   DOM followers totalPrice
##   <fct> <fct> <dbl> <dbl>   <dbl> <fct>     <dbl>     <int>      <dbl>
## 1 http~ 1010~  116.  40.0 1.11e12 2016-08-~  1464       106        415
## 2 http~ 1010~  116.  39.9 1.11e12 2016-07-~   903       126        575
## 3 http~ 1010~  117.  39.9 1.11e12 2016-12-~  1271        48       1030
## 4 http~ 1010~  116.  40.1 1.11e12 2016-09-~   965       138        298.
## 5 http~ 1010~  116.  39.9 1.11e12 2016-08-~   927       286        392
## 6 http~ 1010~  116.  40.0 1.11e12 2016-07-~   861        57        276.
## # ... with 17 more variables: price <int>, square <dbl>, livingRoom <fct>,
## #   drawingRoom <fct>, kitchen <int>, bathRoom <fct>, floor <fct>,
## #   buildingType <dbl>, constructionTime <fct>, renovationCondition <int>,
## #   buildingStructure <int>, ladderRatio <dbl>, elevator <dbl>,
## #   fiveYearsProperty <dbl>, subway <dbl>, district <int>,
## #   communityAverage <dbl>
```

2.2 Missing Data

```r
#Tidy Data: Find all the missing Data in variables
missing <- tibble(na = sapply(data, function(x) any(is.na(x) | is.infinite(x))),
                  sum_na = sapply(data, function(x) sum(is.na(x))),
                  name = colnames(data)) %>%
  filter(na == TRUE)
missing
```

```
## # A tibble: 6 x 3
##   na    sum_na name
##   <lgl>  <int> <chr>
## 1 TRUE  157977 DOM
## 2 TRUE    2021 buildingType
## 3 TRUE      32 elevator
## 4 TRUE      32 fiveYearsProperty
## 5 TRUE      32 subway
## 6 TRUE     463 communityAverage
```

As we can see 6 variables contain missing data, with the according amount. Important to notice is that, almost 50 % of the variable of DOM are missing, hence we cannot just drop the missing observations from this particular variable, but from the others. Further, we are looking into DOM, to determine how to replace the big amount of NA's.

```r
# # Tidy Data: Drop all the missing Data except DOM
# data <- data %>% drop_na(-DOM)
#
# plot(density(data$DOM, na.rm = T))
```

2.3 Data Cleaning

As we know, the variable DOM has lots of missing values, hence we follow the siggestion of Mr. Bouchet and replace the missing values with the median. Furthermore, we extract the floor number of the wrongly imported floor variable.

```r
# Replace NaN of DOM with median and change certian variables as numeric.
data %>% select(c(tradeTime, totalPrice, price, square)) %>% summary(.)
```

```
##       tradeTime        totalPrice          price            square
##  2016-02-28:  1096   Min.   :    0.1   Min.   :     1   Min.   :   6.90
##  2016-03-06:   948   1st Qu.:  205.0   1st Qu.: 28050   1st Qu.:  57.90
##  2016-07-31:   940   Median :  294.0   Median : 38737   Median :  74.26
##  2016-08-31:   910   Mean   :  349.0   Mean   : 43530   Mean   :  83.24
##  2016-03-05:   824   3rd Qu.:  425.5   3rd Qu.: 53820   3rd Qu.:  98.71
##  2016-08-29:   823   Max.   :18130.0   Max.   :156250   Max.   :1745.50
##  (Other)   :313310
```

```r
data <- mutate(data,
               url = as.character(url),
               id = as.character(id),
               followers = as.numeric(followers),
               price = as.numeric(price))

# change rooms
data %>% select(c(floor, livingRoom, drawingRoom, kitchen, bathRoom)) %>% summary(.)
```

```
##      floor           livingRoom       drawingRoom        kitchen
##  ÖÐ 6    : 34788   2       :160589   1       :225659   Min.   :0.0000
##  ¶¥ 6    : 22763   1       : 82386   2       : 72502   1st Qu.:1.0000
##  ¸ß 6    : 20904   3       : 67611   0       : 19686   Median :1.0000
##  µÍ 6    : 15737   4       :  6821   3       :   918   Mean   :0.9946
##  µ× 6    : 13338   5       :  1107   4       :    47   3rd Qu.:1.0000
##  ÖÐ 5    :  8227   6       :   228   µÍ 6    :     7   Max.   :4.0000
##  (Other):203094   (Other):   109   (Other):    32
##     bathRoom
##  1       :261488
##  2       : 52606
##  3       :  3240
##  0       :   915
```

```
## 4       :   489
## 5       :    69
## (Other):    44
```

```
data <- mutate(data,
               floor = as.numeric((str_extract(floor, "[0-9]+"))),
               livingRoom = as.numeric((str_extract(livingRoom, "[0-9]+"))),
               drawingRoom = as.numeric((str_extract(drawingRoom, "[0-9]+"))),
               kitchen = as.numeric((str_extract(kitchen, "[0-9]+"))),
               bathRoom = as.numeric((str_extract(bathRoom, "[0-9]+"))))
```

2.4 Categorical Variable Adjustment

Now we classify each categorical variable into the according group (levels/factors)

```
data %>% select(c(buildingType, constructionTime, renovationCondition, buildingStructure,
                  elevator, fiveYearsProperty, ladderRatio)) %>% summary(.)
```

```
##  buildingType   constructionTime renovationCondition buildingStructure
## Min.   :0.048   2004   : 21145   Min.   :0.000       Min.   :0.000
## 1st Qu.:1.000   2003   : 19409   1st Qu.:1.000       1st Qu.:2.000
## Median :4.000   Î´Öª   : 19283   Median :3.000       Median :6.000
## Mean   :3.010   2005   : 18924   Mean   :2.606       Mean   :4.451
## 3rd Qu.:4.000   2006   : 14854   3rd Qu.:4.000       3rd Qu.:6.000
## Max.   :4.000   2007   : 14213   Max.   :4.000       Max.   :6.000
## NA's   :2021    (Other):211023
##    elevator     fiveYearsProperty  ladderRatio
## Min.   :0.000   Min.   :0.0000    Min.   :       0
## 1st Qu.:0.000   1st Qu.:0.0000    1st Qu.:       0
## Median :1.000   Median :1.0000    Median :       0
## Mean   :0.577   Mean   :0.6456    Mean   :      63
## 3rd Qu.:1.000   3rd Qu.:1.0000    3rd Qu.:       0
## Max.   :1.000   Max.   :1.0000    Max.   :10009400
## NA's   :32      NA's   :32
```

```
# Generate Grouping-Functions:

# Buildingtype names
makeBuildingType <- function(x){
  if(!is.na(x)){
    if(x==1){
      return('Tower')
    }
    else if (x==2){
      return('Bungalow')
    }
```

```r
    else if (x==3){
      return('Mix_plate_tower')
    }
    else if (x==4){
      return('plate')
    }
    else return('wrong_coded')
  }
  else{return('missing')}
}

# Renovationcondition Names
makeRenovationCondition <- function(x){
  if(x==1){
    return('Other')
  }
  else if (x==2){
    return('Rough')
  }
  else if (x==3){
    return('Simplicity')
  }
  else if (x==4){
    return('Hardcover')
  }
  else{return('missing')}
}

# Buldingstructure Names
makeBuildingStructure <- function(x){
  if(x==1){
    return('Unknown')
  }
  else if (x==2){
    return('Mix')
  }
  else if (x==3){
    return('Brick_Wood')
  }
  else if (x==4){
    return('Brick_Concrete')
  }
  else if (x==5){
    return('Steel')
  }
  else if (x==6){
    return('Steel_Concrete')
```

```r
  }
  else{return('missing')}
}

# make District names
makeDistrict <- function(x){
  if(!is.na(x)){
    if(x==1){
      return('Dong Cheng')
    }
    else if (x==2){
      return('Chong Wen & Xuan Wu')
    }
    else if (x==3){
      return('Feng Tai')
    }
    else if (x==4){
      return('Da Xing')
    }
    else if (x==5){
      return('Fang Shan')
    }
    else if (x==6){
      return('Chang Ping')
    }
    else if (x==7){
      return('Chao Yang')
    }
    else if (x==8){
      return('Hai Dian')
    }
    else if (x==9){
      return('Shi Jing Shan')
    }
    else if (x==10){
      return('Xi Cheng')
    }
    else if (x==11){
      return('Tong Zhou')
    }
    else if (x==12){
      return('Men Tou Gou')
    }
    else if (x==13){
      return('Shun Yi')
    }
    else return('wrong_coded')
```

```r
  }
  else{return('missing')}
}

# Mutate rest of the Variables into categorical variables:
data <- mutate(data,
               buildingType = sapply(buildingType, makeBuildingType),
               renovationCondition = as.factor(sapply(renovationCondition,
                                                      makeRenovationCondition)),
               buildingStructure = sapply(buildingStructure,
                                          makeBuildingStructure),
               subway = ifelse(subway == 1, 'has_subway', 'no_subway'),
               fiveYearsProperty = ifelse(fiveYearsProperty == 1,
                                          'owner_less_5y', 'owner_more_5y'),
               elevator = ifelse(elevator == 1, 'has_elevator' , 'no_elevator'),
               district = sapply(district, makeDistrict))

# change building related attributes
data <- mutate(data,
               buildingType = as.factor(buildingType),
               buildingStructure = as.factor(buildingStructure),
               elevator = as.factor(elevator),
               fiveYearsProperty = as.factor(fiveYearsProperty),
               ladderRatio = as.numeric(ladderRatio),
               renovationCondition = as.factor(renovationCondition),
               subway = as.factor(subway),
               district = as.factor(district))

missing2 <- tibble(na = sapply(data, function(x) any(is.na(x) | is.infinite(x))),
                   sum_na = sapply(data, function(x) sum(is.na(x))),
                   name = colnames(data)) %>%
  filter(na == TRUE)
missing2
```

```
## # A tibble: 8 x 3
##   na    sum_na name
##   <lgl>  <int> <chr>
## 1 TRUE  157977 DOM
## 2 TRUE      32 livingRoom
## 3 TRUE       2 bathRoom
## 4 TRUE      32 floor
## 5 TRUE      32 elevator
## 6 TRUE      32 fiveYearsProperty
## 7 TRUE      32 subway
## 8 TRUE     463 communityAverage
```

As we can see some missing data appeared as we set our categorical variables and separated some of the variables.

To make use of the time stamp, we generate the floor dates of the year, month and day. Additionaly calculate the weekday on which the most "transactions" happen (in our case taken offline from the webpage).

To incorporate the geo information of each object. As we have the longitude and latitude coordinates, we can calculate the distance for each object from the city center of Beijing. The coordinates of the city center are followed by the webpage wikipedia:

39.9042º N, 116.4074º E

```
## Adjusting the time variables
# Declare tradeTime as Date,
# extract floor dates: year, month, day
# Adjust constructionTime and generate buldingAge
data <- mutate(data,
               tradeTime = as_datetime(tradeTime),
               tradeYear = floor_date(tradeTime, unit = "year"),
               tradeMonth = floor_date(tradeTime, unit = "month"),
               tradeDay = floor_date(tradeTime, unit = "day"),
               tradeDays = as.numeric(format(tradeTime, format="%d")),
               constructionTime = as.numeric((str_extract(constructionTime, "[0-9]+"))))

# Distance calculation via harversine

bj_lat <- 39.9042
bj_log <- 116.4074

data <- data %>%
  mutate(distance = distHaversine(cbind(Lng, Lat), cbind(bj_log, bj_lat), r=6378137))

missing3 <- tibble(na = sapply(data, function(x) any(is.na(x) | is.infinite(x))),
                   sum_na = sapply(data, function(x) sum(is.na(x))),
                   name = colnames(data)) %>%
  filter(na == TRUE)
missing3
```

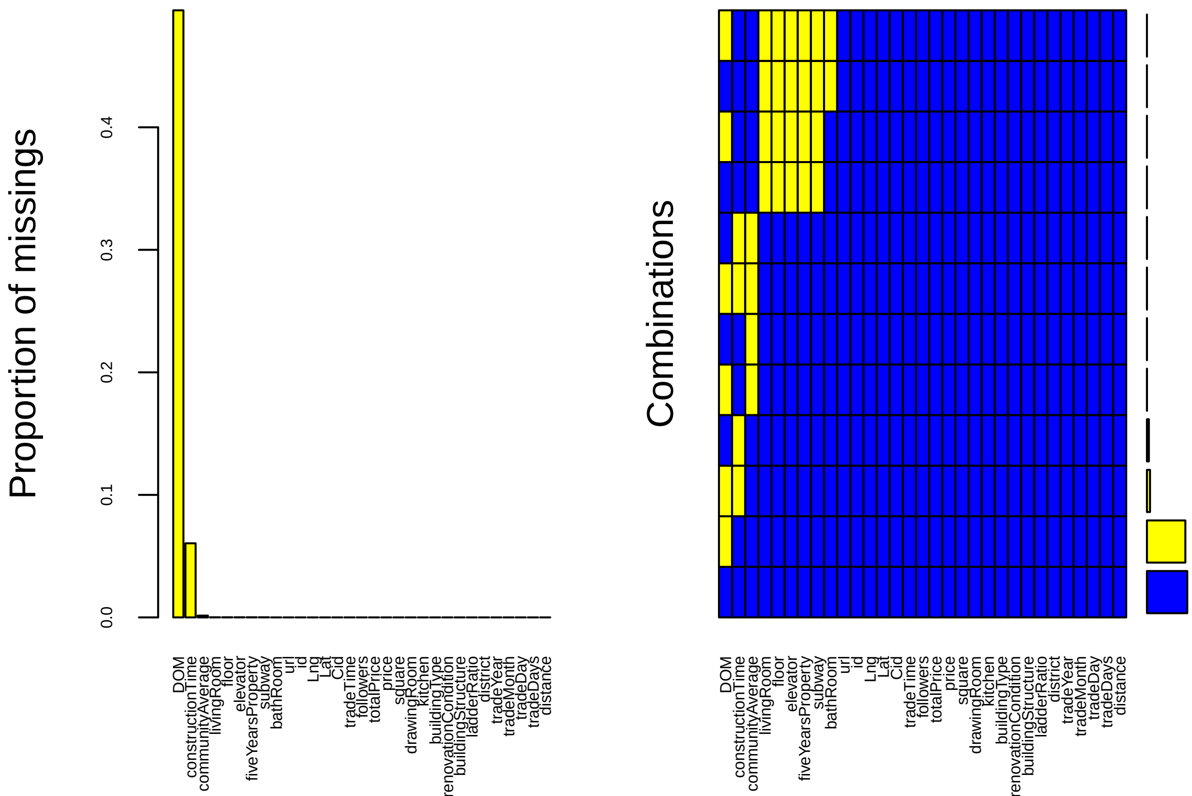
```
## # A tibble: 9 x 3
##   na    sum_na name
##   <lgl>  <int> <chr>
## 1 TRUE  157977 DOM
## 2 TRUE      32 livingRoom
## 3 TRUE       2 bathRoom
## 4 TRUE      32 floor
## 5 TRUE   19283 constructionTime
## 6 TRUE      32 elevator
## 7 TRUE      32 fiveYearsProperty
## 8 TRUE      32 subway
## 9 TRUE     463 communityAverage
```

2.4.1 Missing variables treatment

As we have seen in 'missing3' there are several variables with missing values, lets see whether there is a pattern behind the missing values

```r
# na_pattern <- md.pattern(data)
aggr(data, col = c('blue', 'yellow'),
     numbers = T, sortVars = T,
     labels = names(data), cex.axis = 0.5)
```

```
## Warning in plot.aggr(res, ...): not enough horizontal space to display
## frequencies
```




```
##
##  Variables sorted by number of missings:
##             Variable        Count
##                  DOM 4.954571e-01
##     constructionTime 6.047652e-02
##     communityAverage 1.452089e-03
##           livingRoom 1.003604e-04
##                floor 1.003604e-04
##             elevator 1.003604e-04
```

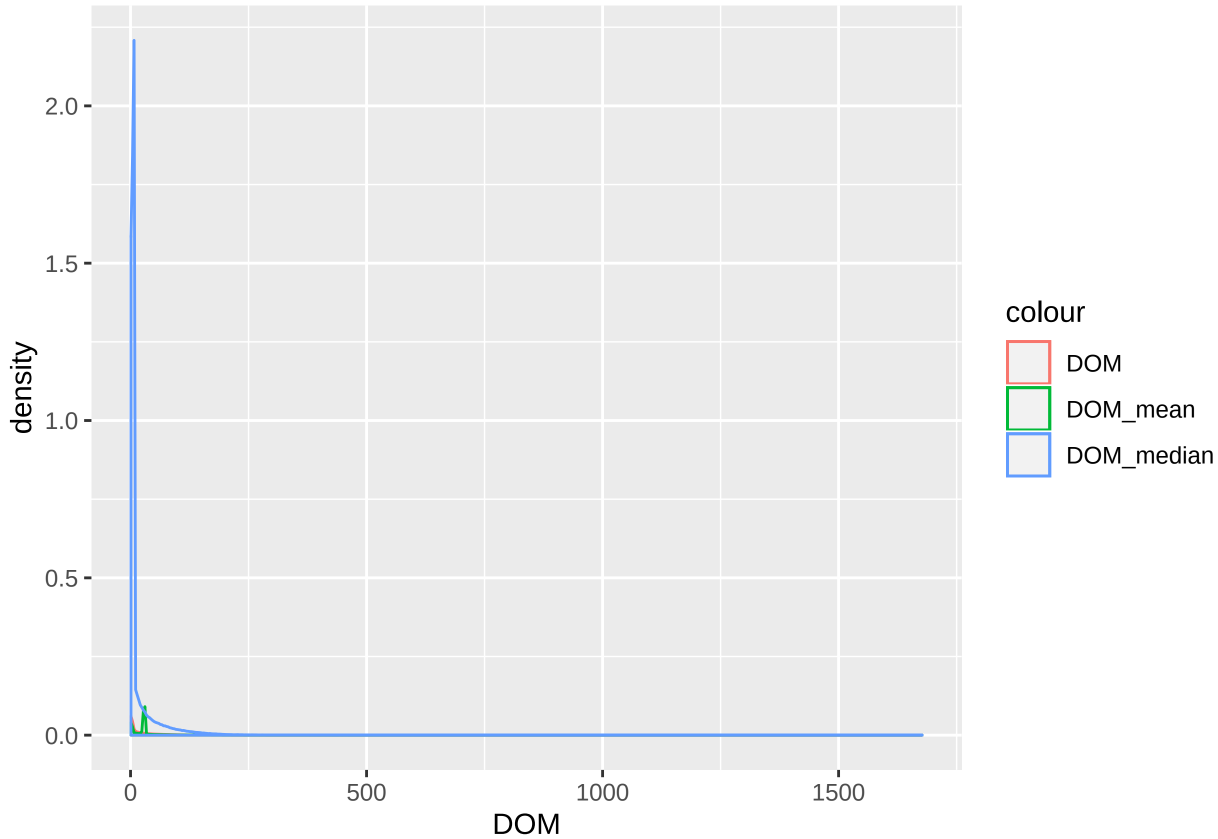
```
##    fiveYearsProperty 1.003604e-04
##               subway 1.003604e-04
##             bathRoom 6.272522e-06
##                  url 0.000000e+00
##                   id 0.000000e+00
##                  Lng 0.000000e+00
##                  Lat 0.000000e+00
##                  Cid 0.000000e+00
##            tradeTime 0.000000e+00
##            followers 0.000000e+00
##           totalPrice 0.000000e+00
##                price 0.000000e+00
##               square 0.000000e+00
##          drawingRoom 0.000000e+00
##              kitchen 0.000000e+00
##         buildingType 0.000000e+00
##  renovationCondition 0.000000e+00
##    buildingStructure 0.000000e+00
##          ladderRatio 0.000000e+00
##             district 0.000000e+00
##            tradeYear 0.000000e+00
##           tradeMonth 0.000000e+00
##             tradeDay 0.000000e+00
##            tradeDays 0.000000e+00
##             distance 0.000000e+00
```

Given the plot of the aggregated data, there is hardly any pattern of missing data.

Because of the vast options of treatments, I will first compare mean and median imputation for DOM and constructionTime. Other variable seem relatively insigificant, as their numbers are in the permille level.

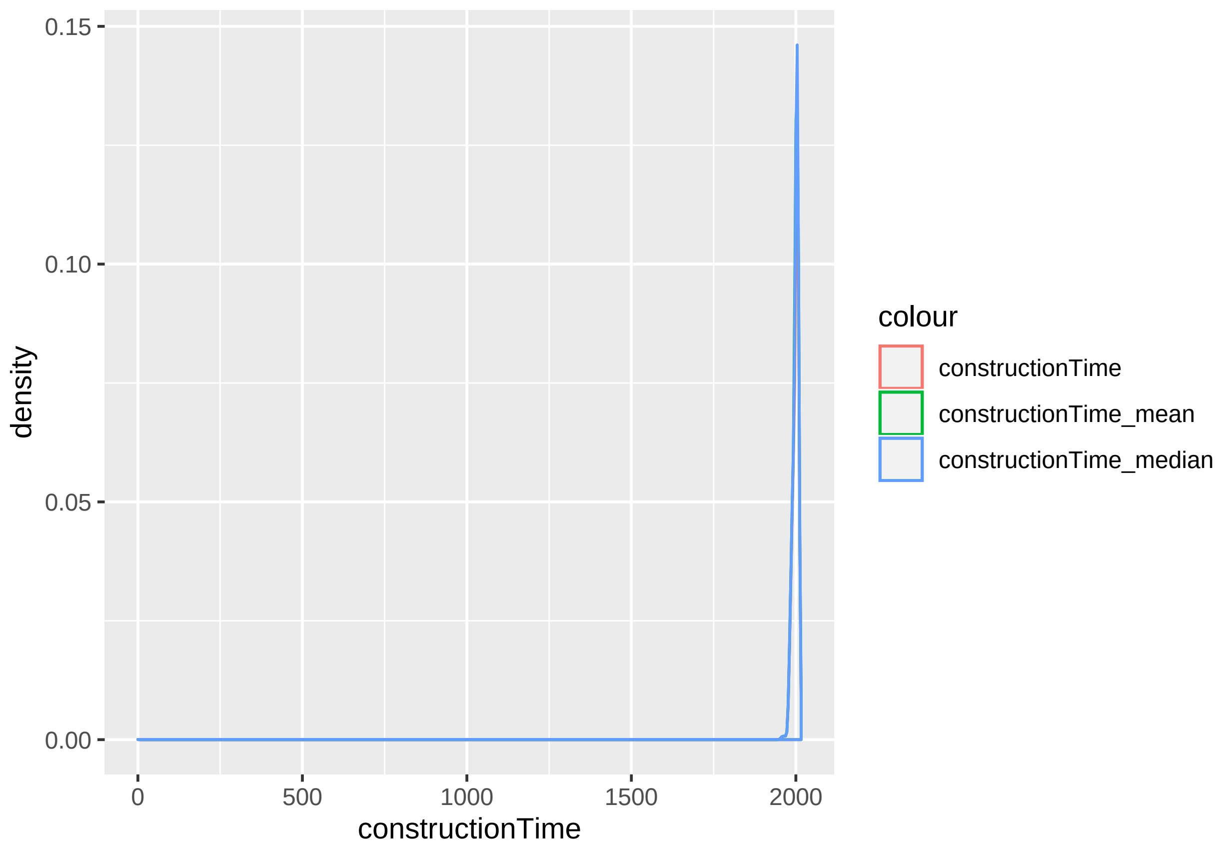
```
data2<- mutate(data,
               DOM_mean = ifelse(is.na(DOM), mean(DOM,na.rm=T), DOM),
               DOM_median = ifelse(is.na(DOM), median(DOM,na.rm=T), DOM),
               constructionTime_mean = ifelse(is.na(constructionTime),
                                              mean(constructionTime,na.rm=T),
                                              constructionTime),
               constructionTime_median = ifelse(is.na(constructionTime),
                                                median(constructionTime,na.rm=T),
                                                constructionTime))

ggplot(data = data2) +
  geom_density(aes(x = DOM, color = "DOM")) +
  geom_density(aes(x = DOM_mean, color = "DOM_mean")) +
  geom_density(aes(x = DOM_median, color = "DOM_median"))
```

```
## Warning: Removed 157977 rows containing non-finite values (stat_density).
```




```r
ggplot(data = data2) +
  geom_density(aes(x = constructionTime, color = "constructionTime")) +
  geom_density(aes(x = constructionTime_mean, color = "constructionTime_mean")) +
  geom_density(aes(x = constructionTime_median, color = "constructionTime_median"))
```

```
## Warning: Removed 19283 rows containing non-finite values (stat_density).
```



Given the DOM plot, we can see a lot changes from the imputation of mean or median. Within the constructionTime we can see minor changes, which do not create any significant bias. Hence the situation with DOM, we will implement an imputation via 'cart' of the MICE package.

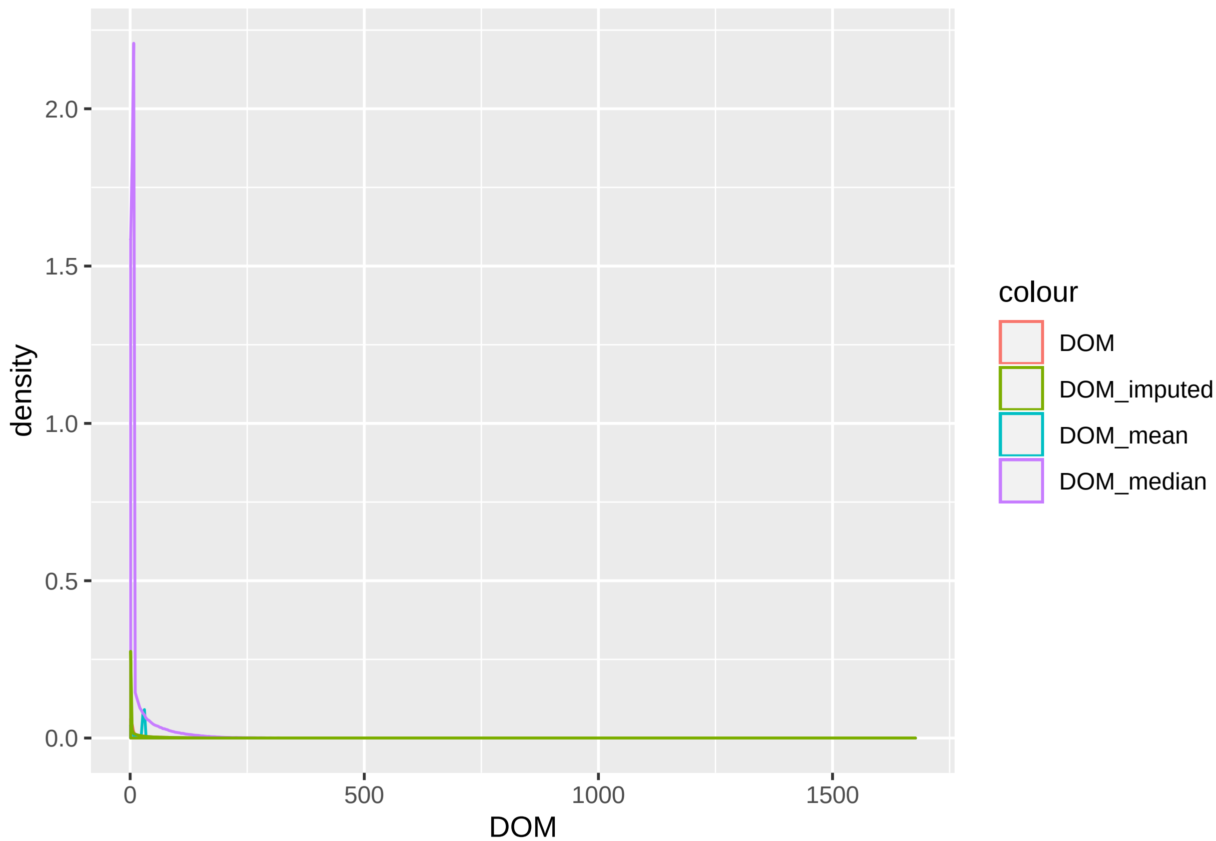
```
# remove irrelevant data and prepare for imputation
data_for_imput <- data %>%
  select(-c(url, Cid, price, followers, ladderRatio, distance, drawingRoom, kitchen,
            tradeTime, tradeYear, tradeMonth, tradeDay, tradeDays)) %>%
  as.data.frame()

# initiate imputation
start_mice <- Sys.time()
imputed <- mice.par(data_for_imput, m = 1, maxit = 5, method = 'cart', seed = 123)
time_mice <- Sys.time() - start_mice
time_mice
```

```
## Time difference of 48.03947 mins
```

```
# extract
imputed_data <- complete(imputed, 1)

# combine
extracted <- select(data,
```

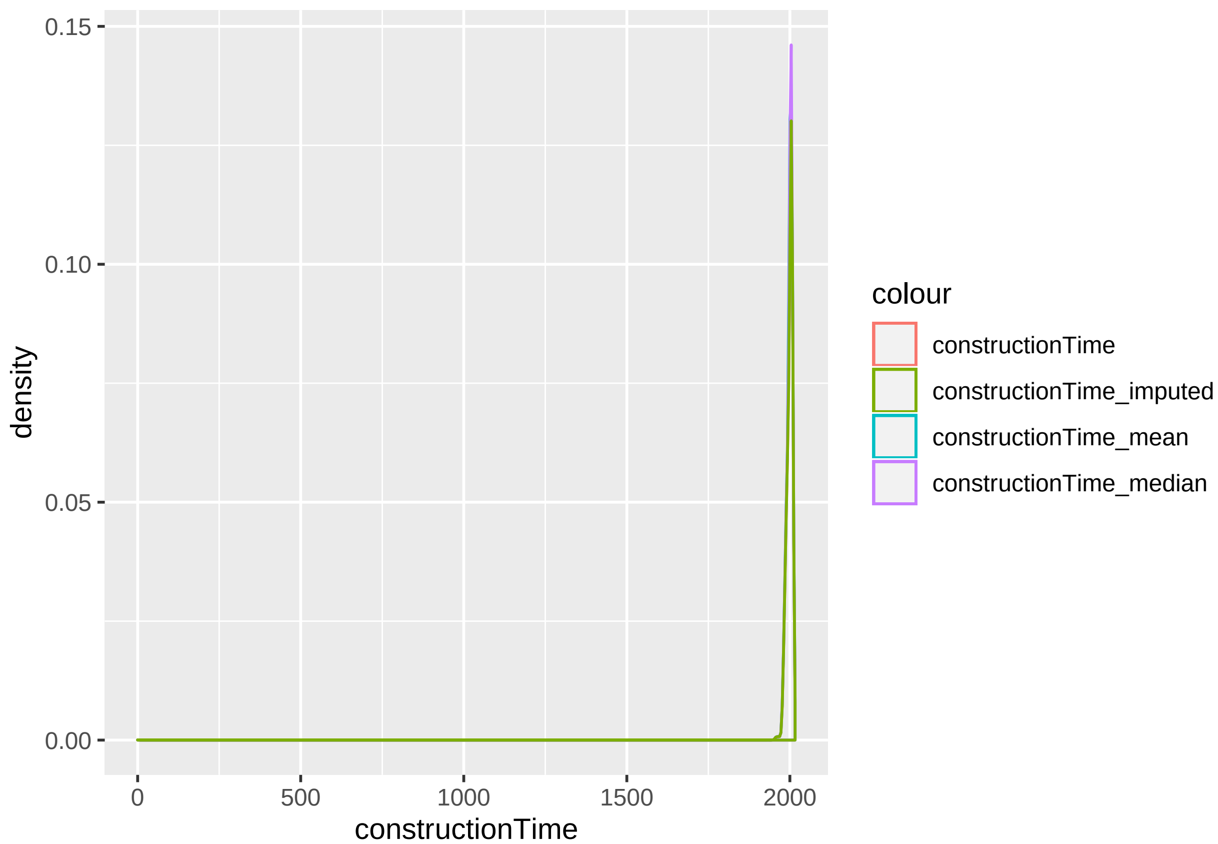
```r
                c(url, id, Cid, price, followers, ladderRatio, distance, drawingRoom,
                  kitchen, tradeTime, tradeYear, tradeMonth, tradeDay, tradeDays))
data_analysis <- left_join(imputed_data, extracted, by = "id")

ggplot(data = data2) +
  geom_density(aes(x = DOM, color = "DOM")) +
  geom_density(aes(x = DOM_mean, color = "DOM_mean")) +
  geom_density(aes(x = DOM_median, color = "DOM_median")) +
  geom_density(aes(x = data_analysis$DOM, color = "DOM_imputed"))
```

```
## Warning: Removed 157977 rows containing non-finite values (stat_density).
```




```r
ggplot(data = data2) +
  geom_density(aes(x = constructionTime, color = "constructionTime")) +
  geom_density(aes(x = constructionTime_mean,
                   color = "constructionTime_mean")) +
  geom_density(aes(x = constructionTime_median,
                   color = "constructionTime_median")) +
  geom_density(aes(x = data_analysis$constructionTime,
                   color = "constructionTime_imputed"))
```

```
## Warning: Removed 19283 rows containing non-finite values (stat_density).
```




```
tibble(na = sapply(data_analysis, function(x) any(is.na(x) | is.infinite(x))),
                   sum_na = sapply(data_analysis, function(x) sum(is.na(x))),
                   name = colnames(data_analysis)) %>%
  filter(na == TRUE)
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: na <lgl>, sum_na <int>, name <chr>
```

As shown by the last two plots, the imputed values look much smoother, compared to the mean/median solution. And all missing values have been replaced.

3 Exploratory Data Analysis

3.1 Summary

Summarize all key variables

```
# Summary of key variables, excluding dates and factors.
df_sum <- data_analysis %>%
  select(-c(url, id, Lng, Lat, Cid, tradeTime, buildingType, constructionTime,
            renovationCondition, buildingStructure, elevator, fiveYearsProperty,
```

```r
            subway, district)) %>%
        summarise_each(funs(min = min,
                          q25 = quantile(., 0.25),
                          median = median,
                          q75 = quantile(., 0.75),
                          max = max,
                          mean = mean,
                          sd = sd))

df_sum <- df_sum %>% gather(stat, val) %>%
                 separate(stat, into = c("var", "stat"), sep = "_") %>%
                 spread(stat, val) %>%
  rename(name = var)

df_sum
```

```
##                name          max         mean       median          min
## 1          bathRoom 2.011000e+03 1.389016e+00 1.000000e+00 0.000000e+00
## 2  communityAverage 1.831090e+05 6.371927e+04 5.901500e+04 1.084700e+04
## 3          distance 4.544371e+04 1.297782e+04 1.131013e+04 4.484510e+02
## 4               DOM 1.677000e+03 1.936896e+01 1.000000e+00 1.000000e+00
## 5       drawingRoom 2.800000e+01 1.172971e+00 1.000000e+00 0.000000e+00
## 6             floor 6.300000e+01 1.329919e+01 1.100000e+01 1.000000e+00
## 7         followers 1.143000e+03 1.673151e+01 5.000000e+00 0.000000e+00
## 8           kitchen 4.000000e+00 9.945994e-01 1.000000e+00 0.000000e+00
## 9       ladderRatio 1.000940e+07 6.316486e+01 3.330000e-01 0.000000e+00
## 10       livingRoom 9.000000e+00 2.010416e+00 2.000000e+00 0.000000e+00
## 11            price 1.562500e+05 4.353044e+04 3.873700e+04 1.000000e+00
## 12           square 1.745500e+03 8.324060e+01 7.426000e+01 6.900000e+00
## 13       totalPrice 1.813000e+04 3.490302e+02 2.940000e+02 1.000000e-01
## 14         tradeDay 1.517098e+09 1.429304e+09 1.442016e+09 1.022890e+09
## 15        tradeDays 3.100000e+01 1.652951e+01 1.700000e+01 1.000000e+00
## 16       tradeMonth 1.514765e+09 1.427962e+09 1.441066e+09 1.022890e+09
## 17        tradeYear 1.514765e+09 1.413257e+09 1.420070e+09 1.009843e+09
##             q25          q75           sd
## 1  1.000000e+00 1.000000e+00 2.005707e+01
## 2  4.633900e+04 7.599300e+04 2.236339e+04
## 3  7.007597e+03 1.783249e+04 7.372154e+03
## 4  1.000000e+00 1.700000e+01 4.503087e+01
## 5  1.000000e+00 1.000000e+00 5.357073e-01
## 6  6.000000e+00 1.900000e+01 7.826744e+00
## 7  0.000000e+00 1.800000e+01 3.420918e+01
## 8  1.000000e+00 1.000000e+00 1.096089e-01
## 9  2.500000e-01 5.000000e-01 2.506851e+04
## 10 1.000000e+00 2.000000e+00 7.768414e-01
## 11 2.805000e+04 5.381950e+04 2.170902e+04
## 12 5.790000e+01 9.871000e+01 3.723466e+01
```

```
## 13 2.050000e+02 4.255000e+02 2.307808e+02
## 14 1.385770e+09 1.469923e+09 5.167998e+07
## 15 9.000000e+00 2.400000e+01 8.741073e+00
## 16 1.383264e+09 1.467331e+09 5.164063e+07
## 17 1.356998e+09 1.451606e+09 5.227888e+07
```

3.2 Frequency Plots

```r
## Monthly

# check how many transactions per month
cat("Complete Set:")
```

```
## Complete Set:
```

```r
table(data_analysis$tradeYear)
```

```
##
## 2002-01-01 2003-01-01 2008-01-01 2009-01-01 2010-01-01 2011-01-01
##          3          1          1          1        189       6010
## 2012-01-01 2013-01-01 2014-01-01 2015-01-01 2016-01-01 2017-01-01
##      37221      38751      32602      69805      90829      43217
## 2018-01-01
##        221
```

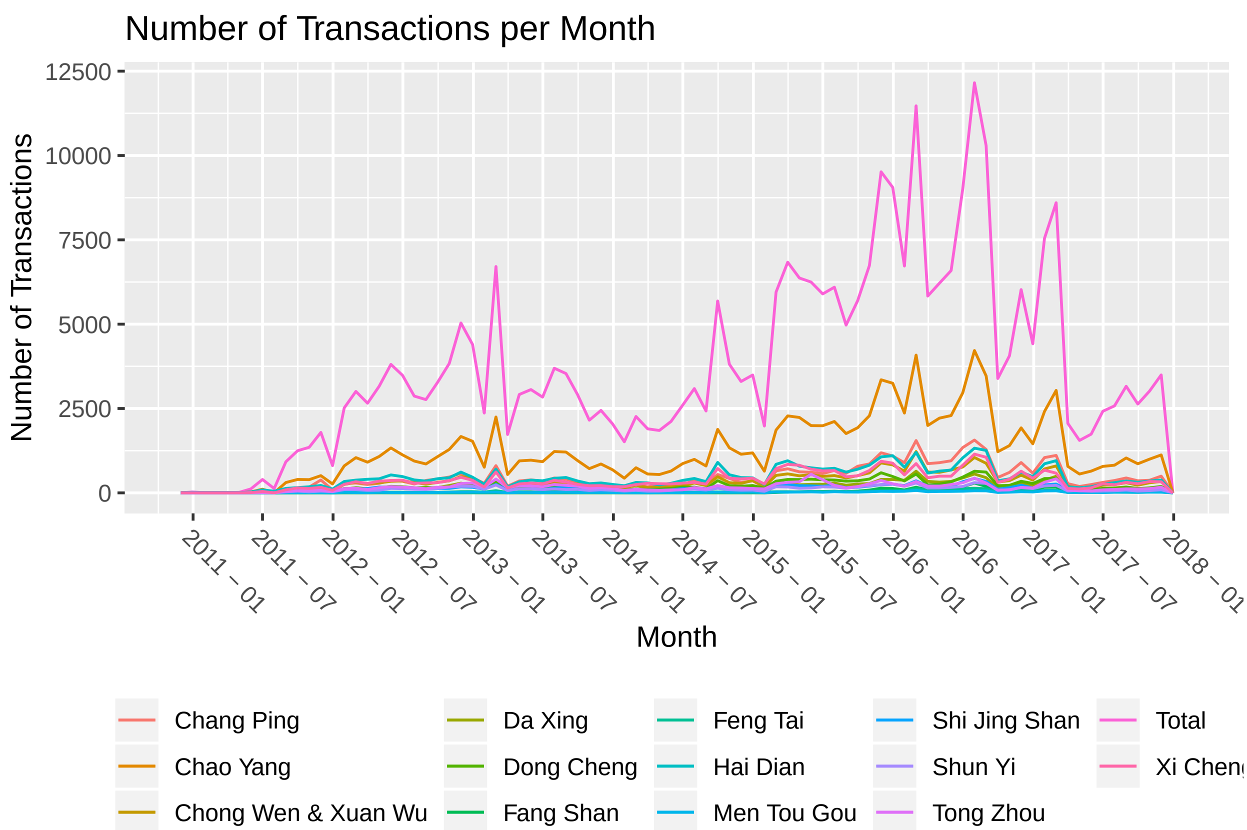
```r
data_analysis <- data_analysis %>%
  filter(tradeYear > "2011-01-01" & tradeYear < "2018-01-01") %>%
  filter(tradeMonth > "2011-01-01" & tradeMonth < "2018-01-01")
cat("\nTruncated Set:")
```

```
##
## Truncated Set:
```

```r
table(data_analysis$tradeYear)
```

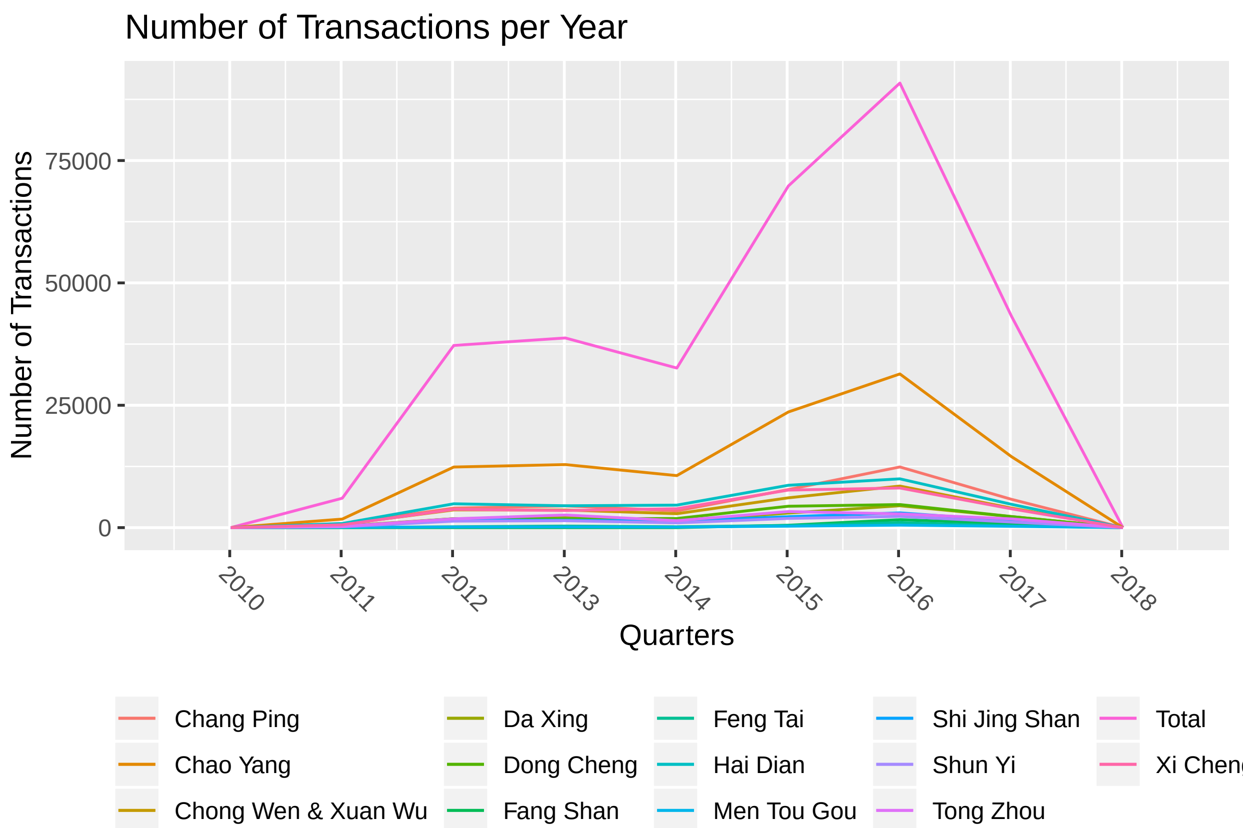
```
##
## 2011-01-01 2012-01-01 2013-01-01 2014-01-01 2015-01-01 2016-01-01
##       6010      37221      38751      32602      69805      90829
## 2017-01-01
##      43217
```

```r
ggplot(data = data_analysis) +
  geom_freqpoly(aes(x = as_date(tradeMonth), color = "Total"), bins = 84) +
  geom_freqpoly(aes(x = as_date(tradeMonth), color = district), bins = 84) +
  scale_x_date(date_breaks = "6 month", date_labels = "%Y - %m") +
  #coord_flip() +
  theme(axis.text.x = element_text(angle = -45,
                                   vjust = 1,
                                   hjust = 0),
        legend.title = element_blank(),
        legend.position = "bottom") +
  labs(title = "Number of Transactions per Month",
       y = "Number of Transactions",
       x = "Month")
```




```r
## Yearly
ggplot(data = data_analysis) +
  geom_freqpoly(aes(x = as_date(tradeYear), color = "Total"), bins = 7) +
  geom_freqpoly(aes(x = as_date(tradeYear), color = district), bins = 7) +
  scale_x_date(date_breaks = "year", date_labels = "%Y") +
  #coord_flip() +
  theme(axis.text.x = element_text(angle = -45,
                                   vjust = 1,
```

```
                                                  hjust = 0),
         legend.title = element_blank(),
         legend.position = "bottom") +
   labs(title = "Number of Transactions per Year",
        y = "Number of Transactions",
        x = "Quarters")
```

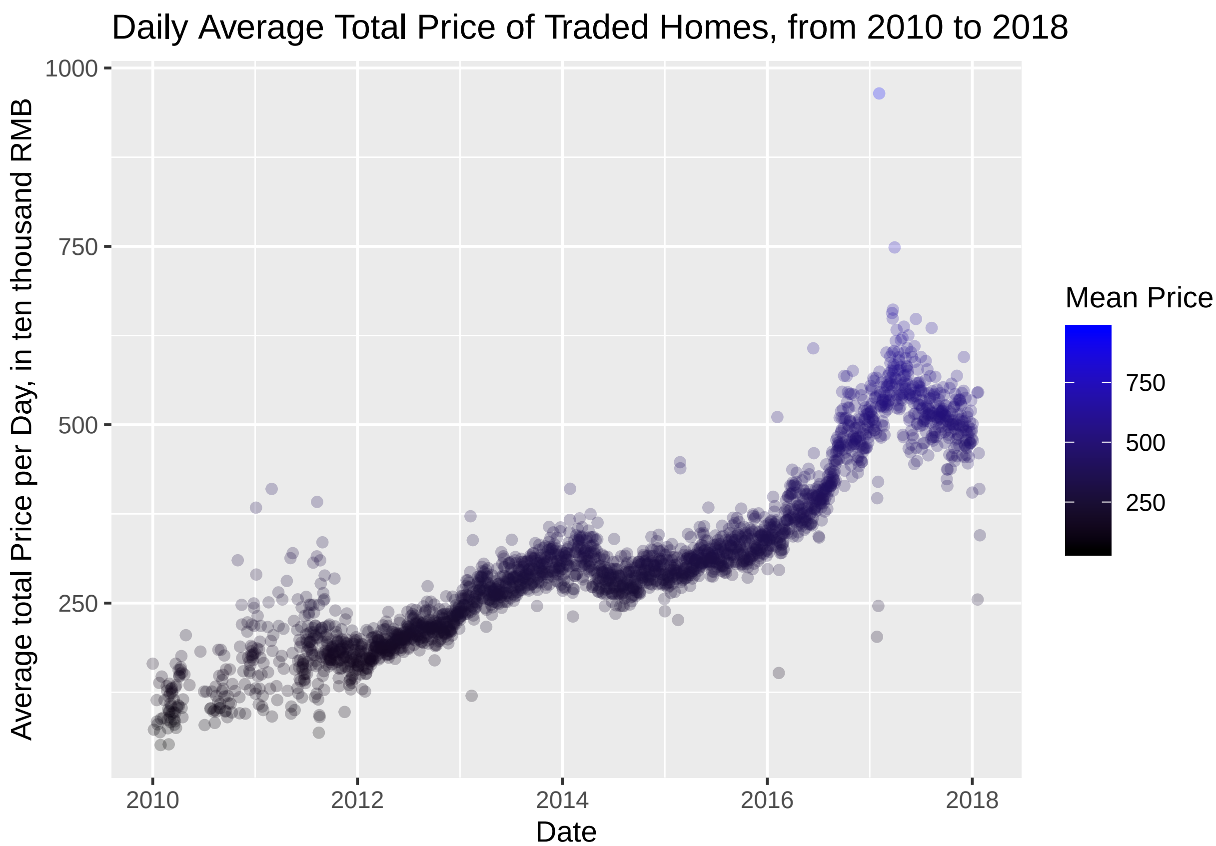


3.3 Plot average daily Price

Now we graph the average daily price. To plot our data more intuitively we omit trades before 2009, as there are only few observations. continuing with saving the plot in the Data priceplot.

```
# Calculate average daily price
avg_price <- data %>% group_by(tradeTime) %>%
                    summarize(mean_price = mean(totalPrice, na.rm = TRUE)) %>%
                    mutate(year = format(tradeTime, format="%Y"))

# Plot average daily price from 2010 on
filter(avg_price, year > 2009) %>%
  ggplot(avg_price, mapping = aes(x = tradeTime, y = mean_price)) +
  geom_point(aes(colour=mean_price), alpha=.25) +
```

```r
  labs(
    title='Daily Average Total Price of Traded Homes, from 2010 to 2018',
    x = "Date",
    y = "Average total Price per Day, in ten thousand RMB",
    colour = "Mean Price") +
  scale_colour_gradient(low = "black", high = "blue1") +
  scale_radius(range=c(1,10))
```




```r
  ggsave("DailyAvg.Price.pdf")
```

```
## Saving 6.5 x 4.5 in image
```

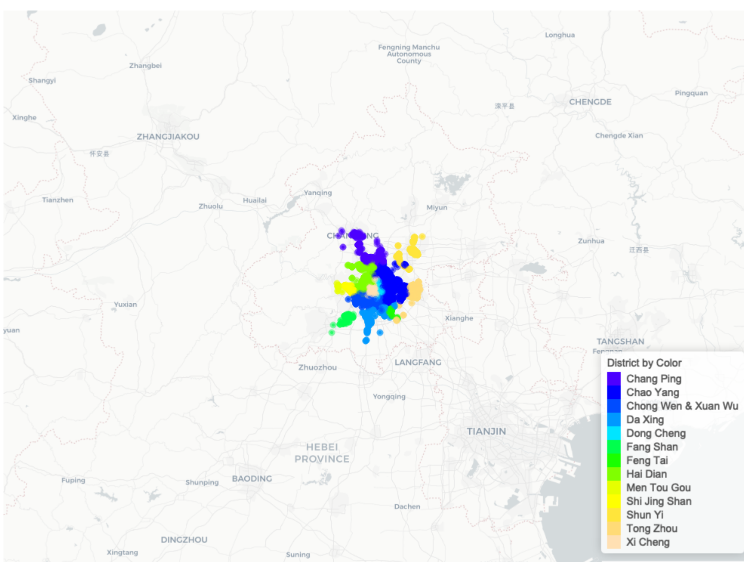
Interesting in this plot is, that we cann see that the average daily price rise over the course of approximately eight years. Although, this is only a rough estimation, as we don't know where and which objects were sold.

3.4 Location Plots

Please note, interactive maps can be found in the HTML file!

```r
data_analysis_sample <- sample_n(data_analysis, 10000, replace = FALSE)

pal <- colorFactor(palette = topo.colors(nlevels(data$district)), domain = data$district[1:13]
map_loc <- leaflet(data_analysis_sample) %>%
  #addTiles() %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  setView(lng=116.4074, lat=39.9042, zoom = 8) %>%
  addCircleMarkers(
    ~Lng, ~Lat,
    radius = 2,
    color = ~pal(district),
    stroke = T,
    fillOpacity = 0.7,
    popup= ~district
    # clusterOptions = markerClusterOptions()
  ) %>%
  addLegend("bottomright", pal = pal, values = ~district,
            title = "District by Color",
            opacity = 1
  )
mapshot(map_loc, file = paste0(getwd(), "/map_loc.png"))
```

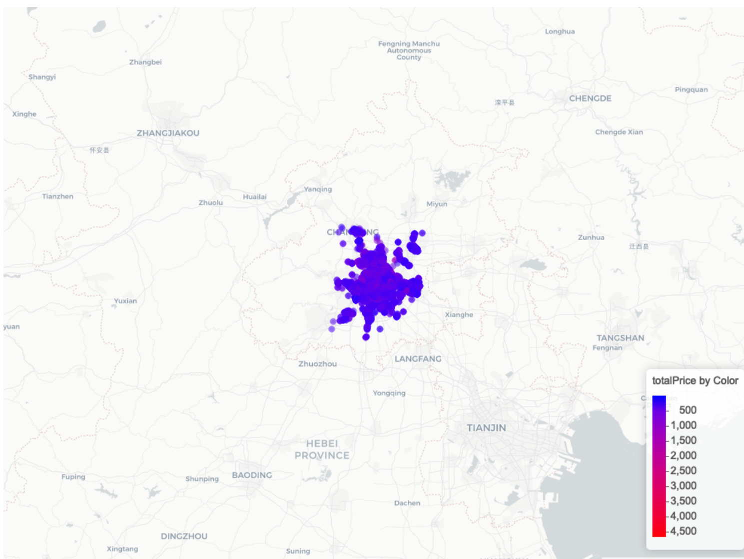
```r
knitr::include_graphics(path = "~/Desktop/BHP2/BHP2/map_loc.png")
```




```r
pal2 <- colorNumeric(c("blue", "red"),
                     domain = data_analysis_sample$totalPrice)
map_price <- leaflet(data_analysis_sample) %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  setView(lng=116.4074, lat=39.9042, zoom = 8) %>%
  addCircleMarkers(
```

```r
    ~Lng, ~Lat,
    radius = 2,
    color = ~pal2(totalPrice),
    #stroke = T,
    #fillOpacity = 0.7,
    popup= ~totalPrice
    # clusterOptions = markerClusterOptions()
  ) %>%
  addLegend("bottomright", pal = pal2, values = ~totalPrice,
            title = "totalPrice by Color",
            opacity = 1
  )
mapshot(map_price, file = paste0(getwd(), "/map_price.png"))
```

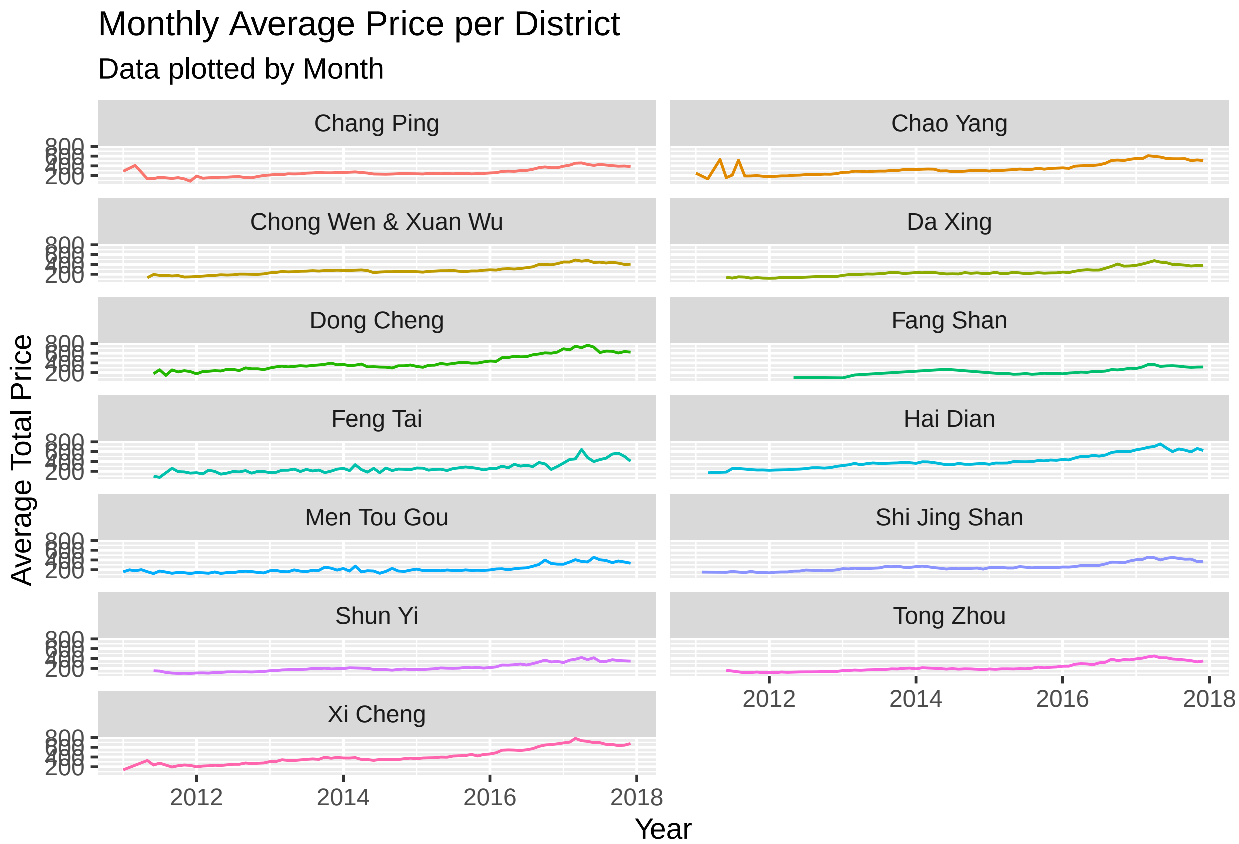
```r
knitr::include_graphics(path = "~/Desktop/BHP2/BHP2/map_price.png")
```



3.5 Price Evolution for each District by Year

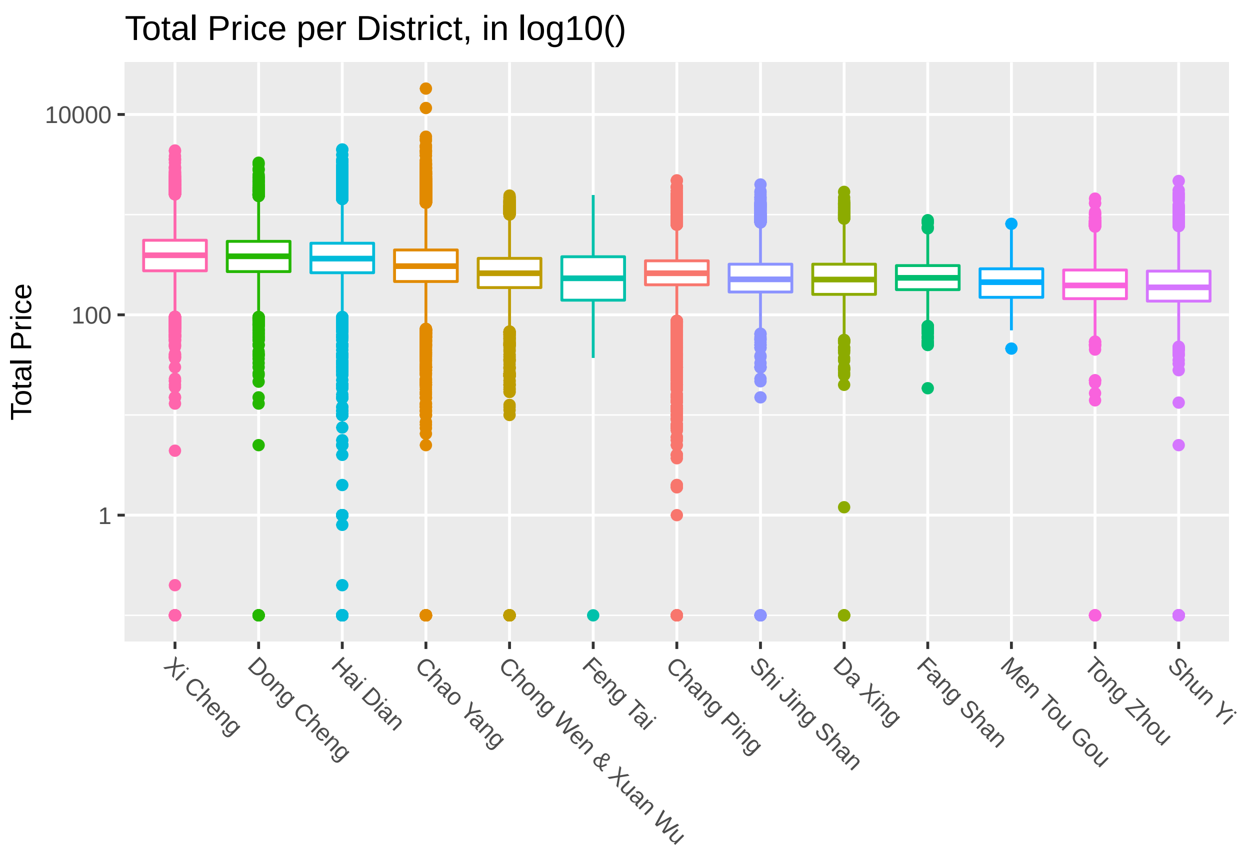
```r
# Monthly Average Price per District
data_analysis %>% filter(tradeYear > 2010) %>%
  group_by(month=floor_date(tradeTime, "month"), district) %>%
  summarize(summary_variable=mean(totalPrice)) %>%
  ggplot(aes(month, summary_variable, color = district)) +
  geom_line() +
  facet_wrap( ~ district, ncol = 2) +
  labs(title = "Monthly Average Price per District",
       subtitle = "Data plotted by Month",
       y = "Average Total Price",
```

```
     x = "Year") +
  theme(legend.position = "none")
```




```
# ggsave("map_avgpriceperdistrict.png")
```

```
# Total Price Change per District (absolute)
data_analysis %>% filter(tradeYear > 2010) %>%
  group_by(tradeYear, district) %>%
  ggplot() +
  geom_boxplot(aes(x = reorder(district, -totalPrice),
               y = totalPrice,
               color = district)) +
  scale_y_log10() +
  theme(axis.text.x = element_text(angle = -45,
                                   vjust = 1,
                                   hjust = 0),
        legend.position = "none",
        axis.title.x = element_blank()) +
#  coord_flip() +
  labs(title = "Total Price per District, in log10()",
     y = "Total Price")
```




```
# ggsave("map_priceperdistrict.png")
```

3.6 Analysis

The nature of this data set is obviously a time series, although I need to say, I have not implmented my time series analysis yet. Furthermore, this analysis is more orientated towards the randomForest part.

3.6.1 TTest

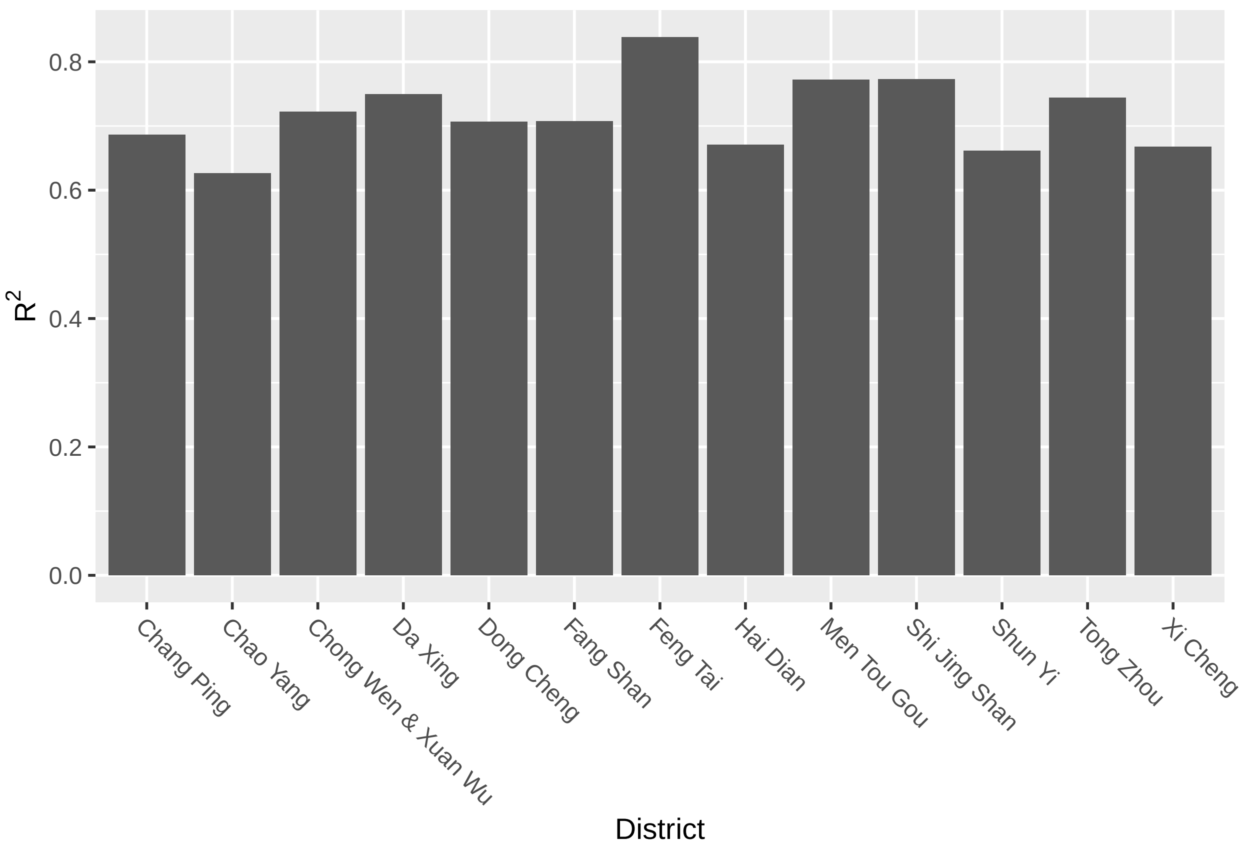
```
ttest <- data_analysis %>%
  nest(-district) %>%
  mutate(fit = map(data, ~t.test(.$totalPrice)),
         p    = map_dbl(fit, "p.value"),
         results = map(fit, glance)) %>%
  unnest(results)
ttest
```

```
## # A tibble: 13 x 12
##    district data  fit        p estimate statistic p.value parameter conf.low
```

```
##    <fct>    <lis> <lis> <dbl>    <dbl>     <dbl>    <dbl>     <dbl>    <dbl>
##  1 Chao Ya~ <tib~ <hte~     0     369.     474.         0    107173     367.
##  2 Chang P~ <tib~ <hte~     0     282.     421.         0     38599     281.
##  3 Dong Ch~ <tib~ <hte~     0     439.     230.         0     17076     436.
##  4 Shun Yi  <tib~ <hte~     0     223.     158.         0      9193     220.
##  5 Xi Cheng <tib~ <hte~     0     449.     301.         0     31275     446.
##  6 Chong W~ <tib~ <hte~     0     297.     320.         0     29316     296.
##  7 Hai Dian <tib~ <hte~     0     429.     319.         0     38171     426.
##  8 Da Xing  <tib~ <hte~     0     258.     227.         0     15299     256.
##  9 Fang Sh~ <tib~ <hte~     0     250.     134.         0      2951     246.
## 10 Feng Tai <tib~ <hte~     0     290.      72.7        0      2535     283.
## 11 Shi Jin~ <tib~ <hte~     0     269.     185.         0     11360     266.
## 12 Men Tou~ <tib~ <hte~     0     236.      82.2        0      1515     231.
## 13 Tong Zh~ <tib~ <hte~     0     229.     218.         0     13959     227.
## # ... with 3 more variables: conf.high <dbl>, method <chr>,
## #   alternative <chr>
```

3.6.2 Linear Regression

```
lin_reg <- data_analysis %>%
  nest(-district) %>%
  mutate(fit = map(data, ~ lm(totalPrice ~
                               DOM +  livingRoom +  drawingRoom +  kitchen +
                               bathRoom + floor +  buildingType +
                               buildingStructure + elevator +  fiveYearsProperty +
                               subway +  factor(tradeYear) + distance + followers,
                             data = .)),
         glance = map(fit, glance),
         augment = map(fit, augment),
         tidy = map(fit, tidy))

lin_reg %>% unnest(glance) %>%
ggplot(data = ) +
  geom_bar(aes(x = factor(district), y = r.squared), stat = "identity") +
  labs(x = "District", y = expression(R^{2})) +
  theme(axis.text.x = element_text(angle = -45, vjust = 1, hjust = 0))
```




```
# ggsave("rsqrtperdistrict.pdf" )
```

As we can see FengTai has the highest R-squared value, lets see how the variables interact with the totalPrice:

```
# Feng Tai
lin_reg_fit <- lin_reg$fit
summary(lin_reg_fit[[10]])
```

```
##
## Call:
## lm(formula = totalPrice ~ DOM + livingRoom + drawingRoom + kitchen +
##     bathRoom + floor + buildingType + buildingStructure + elevator +
##     fiveYearsProperty + subway + factor(tradeYear) + distance +
##     followers, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -363.46  -44.58   -4.74   41.23  672.48
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)                        3.750e+02  3.087e+01  12.149  < 2e-16 ***
## DOM                                2.329e-01  4.211e-02   5.530 3.54e-08 ***
## livingRoom                         9.093e+01  3.151e+00  28.858  < 2e-16 ***
## drawingRoom                        3.672e+01  3.573e+00  10.279  < 2e-16 ***
## kitchen                            4.588e+00  1.731e+01   0.265  0.79099
## bathRoom                           4.588e+01  4.408e+00  10.407  < 2e-16 ***
## floor                              3.298e+00  5.835e-01   5.652 1.76e-08 ***
## buildingTypeMix_plate_tower       -2.624e+02  1.207e+01 -21.738  < 2e-16 ***
## buildingTypeplate                 -1.875e+02  1.070e+01 -17.519  < 2e-16 ***
## buildingTypeTower                 -3.418e+02  1.375e+01 -24.851  < 2e-16 ***
## buildingStructureMix               3.972e+00  8.872e+00   0.448  0.65440
## buildingStructureSteel             3.967e+02  8.285e+01   4.788 1.78e-06 ***
## buildingStructureSteel_Concrete -3.658e+01  8.884e+00  -4.118 3.95e-05 ***
## elevatorno_elevator               -6.082e+01  6.896e+00  -8.820  < 2e-16 ***
## fiveYearsPropertyowner_more_5y     7.272e+00  3.737e+00   1.946  0.05177 .
## subwayno_subway                    2.923e+01  6.471e+00   4.518 6.54e-06 ***
## factor(tradeYear)2012-01-01        1.710e+01  1.771e+01   0.966  0.33432
## factor(tradeYear)2013-01-01        6.824e+01  1.726e+01   3.953 7.94e-05 ***
## factor(tradeYear)2014-01-01        9.049e+01  1.771e+01   5.109 3.48e-07 ***
## factor(tradeYear)2015-01-01        8.609e+01  1.712e+01   5.029 5.28e-07 ***
## factor(tradeYear)2016-01-01        1.544e+02  1.694e+01   9.114  < 2e-16 ***
## factor(tradeYear)2017-01-01        2.816e+02  1.738e+01  16.199  < 2e-16 ***
## distance                          -1.956e-02  9.250e-04 -21.143  < 2e-16 ***
## followers                         -1.964e-01  6.235e-02  -3.150  0.00165 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81.22 on 2512 degrees of freedom
## Multiple R-squared:  0.8386, Adjusted R-squared:  0.8371
## F-statistic: 567.5 on 23 and 2512 DF,  p-value: < 2.2e-16
```

3.6.3 RandomForest Analysis

Now we try the nested randomForest algorithm, to reveal each features variation for each district

```r
# Convenience function to get importance information from a randomForest fit
# into a dataframe
imp_df <- function(rf_fit) {
  imp <- importance(rf_fit)
  vars <- rownames(imp)
  imp %>%
    tibble::as_tibble() %>%
    mutate(var = vars)
}

# Take only 75000 observations as my computing power is limited
data_analysis_sample <- sample_n(data_analysis, size = 75000)
```

```r
set.seed(123)
start_rF <- Sys.time()
rF <- data_analysis_sample %>%
  # Selecting data to work with
  na.omit() %>%
  select(totalPrice, district,
         DOM, livingRoom, drawingRoom, kitchen, bathRoom,
         floor, buildingType, buildingStructure,
         elevator, fiveYearsProperty, subway, tradeYear,
         distance, followers) %>%
  # Nesting data and fitting model
  nest(-district) %>%
  mutate(fit = map(data, ~ randomForest(totalPrice ~ ., data = .,
                                         importance = TRUE,
                                         ntree = 100)),
         importance = map(fit, imp_df)) %>%
  # Unnesting and plotting
  unnest(importance)
time_rF <- Sys.time() - start_rF
time_rF
```

```
## Time difference of 3.735301 mins
```

```r
# Plot each feature and its importance separated with each district to
# understand how each district is different
rFPlot <- ggplot(rF, aes(x = `%IncMSE`, y = var, color = `%IncMSE`)) +
  geom_segment(aes(xend = min(`%IncMSE`), yend = var), alpha = .2) +
  geom_point(size = 3) +
  facet_grid(. ~ district) +
  guides(color = "none") +
  theme_bw() +
  labs(y = "Variable",
       x = "Importance")
```

3.6.4 Gradient Boosting Time Series Analysis

Here we conduct a time series analysis. We take the sum of each trading day and try to predict it.

```r
# Generate a daily time series
data_ts_analysis <- data_analysis %>%
  group_by(tradeDay) %>%
  summarise(totalPrice_sum = sum(totalPrice))

# splitting into train and test
data_ts_analysis_index <- createDataPartition(data_ts_analysis$totalPrice_sum, p = .8, list = F
```

```r
ts_train <- data_ts_analysis[data_ts_analysis_index,]
ts_test <- data_ts_analysis[-data_ts_analysis_index,]

# train control for time slices
myTimeControl <- trainControl(
  method = "timeslice",
  initialWindow = 50,
  horizon = 1,
  fixedWindow = TRUE
)

# train grid for gbm
grid_gbm <- expand.grid(
  n.trees = c(100, 250, 500),
  shrinkage = c(0.001, 0.01),
  interaction.depth = c(1, 16, 20),
  n.minobsinnode = c(1, 2, 4)
)

#find best tune
cl <- makePSOCKcluster(10)
registerDoParallel(cl)
start_gbm <- Sys.time()
gbmts_train <- train(
  totalPrice_sum ~ tradeDay,
  data = ts_train,
  method = "gbm",
  distribution = "gaussian",
  trControl = myTimeControl,
  verbose = FALSE,
  tuneGrid = grid_gbm,
  preProc = c("center", "scale"))
```

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

```r
time_gbm <- Sys.time() - start_gbm
stopCluster(cl)

time_gbm
```

```
## Time difference of 3.445489 mins
```

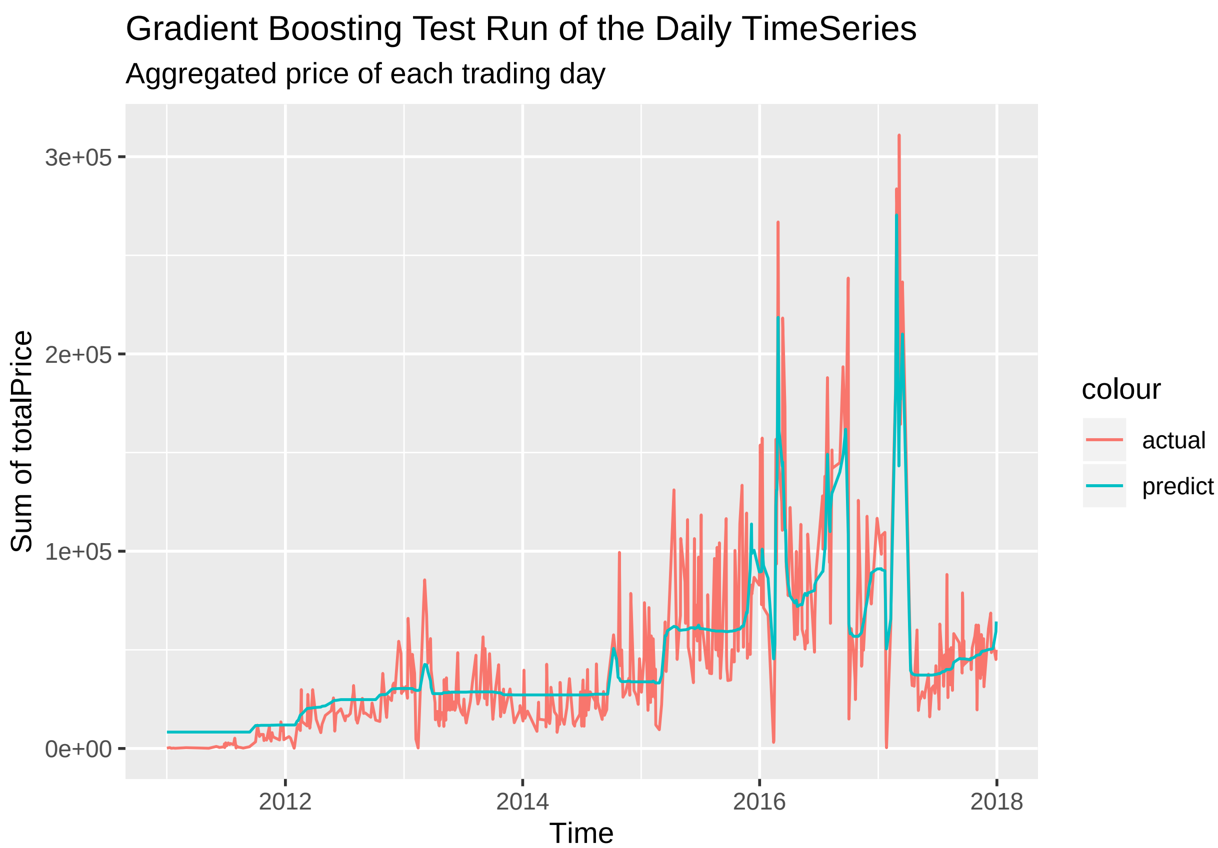
```r
gbmts_train$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 38     250                16      0.01              1
```

```r
# run best tune
cl <- makePSOCKcluster(10)
registerDoParallel(cl)
gbmts_train2 <- train(
  totalPrice_sum ~ tradeDay,
  data = ts_train,
  method = "gbm",
  distribution = "gaussian",
  trControl = myTimeControl,
  verbose = FALSE,
  tuneGrid = gbmts_train$bestTune,
  preProc = c("center", "scale"))
```

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

```r
stopCluster(cl)

# predict values
gbmts_predict <- predict(
  gbmts_train2,
  newdata = ts_test
)

# plot values
ggplot(data = ts_test, aes(x = tradeDay, y = totalPrice_sum)) +
  geom_line(aes(color = "actual")) +
  geom_line(aes(y = gbmts_predict, x = tradeDay,
                color = "predict")) +
  labs(title = "Gradient Boosting Test Run of the Daily TimeSeries",
       subtitle = "Aggregated price of each trading day",
       y = "Sum of totalPrice",
       x = "Time")
```



4 References

- Housing price in Beijing
- Forecasting Beijing's housing prices