# Data Wrangling and randomForest

*Marc Kullmann*

*2019-06-20*

## Contents

## 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 described data file as a dataframe do work with. For convenience, I choose to declare the CSV file as a tibble dataframe.

```
Df <- read.csv('/Users/marckullmann/Desktop/Master 1/R_5811_Paper/new.csv',
               sep=',', fileEncoding="latin1")
data <- as_tibble(Df)
```

Get a quick overview of the underlaying data set.

```
class(data)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
str(data)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    318851 obs. of  26 variables:
##  $ url                : Factor w/ 318851 levels "https://bj.lianjia.com/chengjiao/101084782(
##  $ 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 :
##  $ 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 "¶¥ 6","¸ß 12",..: 6 7 7 6 6 5 6 7 5 5 ...
##  $ 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 "»ìºÏ½á¹¹","¶¥ 10",..: 52 48 125 146 129 131 64
##  $ 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 6(
##  $ 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 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
indx <- as.data.frame(apply(data, 2, function(x) any(is.na(x) | is.infinite(x))),
                      keep.rownames = TRUE) %>%
  rename(value = `apply(data, 2, function(x) any(is.na(x) | is.infinite(x)))`)
indx <- as_tibble(rownames_to_column(indx)) %>%
  mutate(sum_na = sapply(data, function(x) sum(is.na(x)))) %>%
  filter(indx, value == TRUE)
indx
```

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

As we can see 6 variables contain missing data. Now it is interesting how many observations are missing in these 6 variables! Furthermore, we can see almost 50 % of the variable of DOM are missing, hence we cannot drop the missing observations from this particular variable, but from the others.

```r
# Tidy Data: Drop all the missing Data except DOM
data <- data %>% drop_na(-DOM)
```

## 2.3 Classification of Variables

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.

```
# Replace NaN of DOM with median and change certian variables as numeric.
data <- mutate(data, DOM = ifelse(is.na(DOM), median(DOM,na.rm=T), DOM),
               floor = as.numeric((str_extract(floor, "[0-9]+"))),
               followers = as.numeric(followers),
               price = as.numeric(price),
               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]+"))),
               roomNumber = livingRoom + drawingRoom + bathRoom + kitchen
               )
```

```
# Construction Time is labeled as factor, which needs a special treatment

data <- mutate(data, constructionTime =
                 as.numeric((str_extract(constructionTime, "[0-9]+"))))
summary(data$constructionTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1950    1994    2001    1999    2006    2016   18747
```

```
data <- mutate(data, constructionTime = ifelse(is.na(constructionTime),
                                               median(constructionTime,na.rm=T), constructionTi
summary(data$constructionTime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1950    1994    2001    1999    2005    2016
```

Now we have 18747 NA's in our data set. Lets check how constructionTime is distributed after replacing it with the median.

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

```
# Generate Grouping-Functions:
{makeBuildingType <- function(x){
  if(!is.na(x)){
    if(x==1){
      return('Tower')
    }
    else if (x==2){
      return('Bungalow')
    }
    else if (x==3){
      return('Mix_plate_tower')
    }
    else if (x==4){
```

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

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')
  }
}

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')
  }
}
}

# Mutate rest of the Variables into categorical variables:
data <- mutate(data, buildingType = sapply(buildingType, makeBuildingType),
                renovationCondition = 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')
                )

# Finish up the rest of the rest of the categories
data <- mutate(data,
               buildingType = as.factor(buildingType),
               renovationCondition = as.factor(renovationCondition),
               buildingStructure = as.factor(buildingStructure),
               elevator = as.factor(elevator),
               fiveYearsProperty = as.factor(fiveYearsProperty),
               subway = as.factor(subway),
               district = as.factor(district)
               )

# Now we are adjusting the time variable
# Declare tradeTime as Date, extract year, month and day
data <- mutate(data, tradeTime = as.Date(tradeTime, format = "%Y-%m-%d"),
             tradeYear = as.numeric(format(tradeTime, format="%Y")),
             tradeMonth = as.numeric(format(tradeTime, format="%m")),
             tradeDay = as.numeric(format(tradeTime, format="%d")))

# Check the nature of each variable
str(data)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    316448 obs. of  30 variables:
##  $ url                 : Factor w/ 318851 levels "https://bj.lianjia.com/chengjiao/10108478
##  $ 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           : Date, format: "2016-08-09" "2016-07-28" ...
##  $ DOM                 : num  1464 903 1271 965 927 ...
##  $ followers           : num  106 126 48 138 286 57 167 138 218 134 ...
##  $ totalPrice          : num  415 575 1030 298 392 ...
##  $ price               : num  31680 43436 52021 22202 48396 ...
##  $ square              : num  131 132 198 134 81 ...
##  $ livingRoom          : num  2 2 3 3 2 1 2 3 1 1 ...
##  $ drawingRoom         : num  1 2 2 1 1 0 1 2 0 0 ...
##  $ kitchen             : num  1 1 1 1 1 1 1 1 1 0 ...
##  $ bathRoom            : num  1 2 3 1 1 1 1 2 1 0 ...
##  $ floor               : num  26 22 4 21 6 8 6 22 10 23 ...
##  $ buildingType       : Factor w/ 4 levels "Bungalow","Mix_plate_tower",..: 4 4 3 4 3 3 3 4
##  $ constructionTime   : num  2005 2004 2005 2008 1960 ...
##  $ renovationCondition: Factor w/ 4 levels "Hardcover","Other",..: 4 1 4 2 3 4 1 1 2 1 ...
##  $ buildingStructure  : Factor w/ 6 levels "Brick_Concrete",..: 5 5 5 5 3 5 3 5 3 5 ...
##  $ ladderRatio        : num  0.217 0.667 0.5 0.273 0.333 0.333 0.5 0.667 0.333 0.308 ...
##  $ elevator           : Factor w/ 2 levels "has_elevator",..: 1 1 1 1 2 1 2 1 2 1 ...
##  $ fiveYearsProperty  : Factor w/ 2 levels "owner_less_5y",..: 2 1 2 2 1 1 2 1 2 1 ...
##  $ subway             : Factor w/ 2 levels "has_subway","no_subway": 1 2 2 2 1 2 2 2 2 1 ..
##  $ district           : Factor w/ 13 levels "1","2","3","4",..: 7 7 7 6 1 7 7 7 13 1 ...
##  $ communityAverage   : num  56021 71539 48160 51238 62588 ...
##  $ roomNumber         : num  5 7 9 6 5 3 5 8 3 1 ...
##  $ tradeYear          : num  2016 2016 2016 2016 2016 ...
##  $ tradeMonth         : num  8 7 12 9 8 7 7 9 9 9 ...
##  $ tradeDay           : num  9 28 11 30 28 22 14 7 4 5 ...
```

## 2.4 Summary

Summarize all key variables

```
# Summary of key variables, excluding dates and factors.
df.sum <- data %>%
  select(-c(url, id, buildingType, renovationCondition, buildingStructure,
            elevator, fiveYearsProperty, subway, district, tradeTime)) %>%
      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) %>%
                  select(var, min, q25, median, q75, max, mean, sd)
df.sum
```

```
## # A tibble: 20 x 8
##    var              min      q25   median      q75     max     mean       sd
##    <chr>          <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>
##  1 bathRoom    0.        1.00e+ 0 1.00e+ 0 1.00e+ 0 7.00e 0 1.18e+ 0 4.25e- 1
##  2 Cid         1.11e+12 1.11e+12 1.11e+12 1.11e+12 1.18e14 1.13e+12 1.29e+12
##  3 community~ 1.08e+ 4 4.64e+ 4 5.90e+ 4 7.60e+ 4 1.83e 5 6.38e+ 4 2.23e+ 4
##  4 construct~ 1.95e+ 3 1.99e+ 3 2.00e+ 3 2.00e+ 3 2.02e 3 2.00e+ 3 8.51e+ 0
##  5 DOM         1.00e+ 0 6.00e+ 0 6.00e+ 0 7.00e+ 0 1.68e 3 1.74e+ 1 3.73e+ 1
##  6 drawingRo~ 0.        1.00e+ 0 1.00e+ 0 1.00e+ 0 5.00e 0 1.17e+ 0 5.19e- 1
##  7 floor       1.00e+ 0 6.00e+ 0 1.10e+ 1 2.00e+ 1 6.30e 1 1.33e+ 1 7.82e+ 0
##  8 followers   0.        0.        5.00e+ 0 1.80e+ 1 1.14e 3 1.67e+ 1 3.41e+ 1
##  9 kitchen     0.        1.00e+ 0 1.00e+ 0 1.00e+ 0 3.00e 0 9.95e- 1 1.03e- 1
## 10 ladderRat~ 1.40e- 2 2.50e- 1 3.33e- 1 5.00e- 1 1.00e 7 6.36e+ 1 2.52e+ 4
## 11 Lat        3.96e+ 1 3.99e+ 1 3.99e+ 1 4.00e+ 1 4.03e 1 3.99e+ 1 9.12e- 2
## 12 livingRoom 0.       1.00e+ 0 2.00e+ 0 2.00e+ 0 8.00e 0 2.00e+ 0 7.68e- 1
## 13 Lng        1.16e+ 2 1.16e+ 2 1.16e+ 2 1.16e+ 2 1.17e 2 1.16e+ 2 1.12e- 1
## 14 price      1.00e+ 0 2.81e+ 4 3.88e+ 4 5.39e+ 4 1.56e 5 4.35e+ 4 2.17e+ 4
## 15 roomNumber 1.00e+ 0 4.00e+ 0 5.00e+ 0 6.00e+ 0 1.60e 1 5.35e+ 0 1.43e+ 0
## 16 square     7.37e+ 0 5.79e+ 1 7.42e+ 1 9.85e+ 1 6.40e 2 8.28e+ 1 3.59e+ 1
## 17 totalPrice 1.00e- 1 2.05e+ 2 2.94e+ 2 4.25e+ 2 4.90e 3 3.48e+ 2 2.24e+ 2
## 18 tradeDay   1.00e+ 0 9.00e+ 0 1.70e+ 1 2.40e+ 1 3.10e 1 1.65e+ 1 8.74e+ 0
## 19 tradeMonth 1.00e+ 0 3.00e+ 0 7.00e+ 0 1.00e+ 1 1.20e 1 6.61e+ 0 3.48e+ 0
## 20 tradeYear  2.00e+ 3 2.01e+ 3 2.02e+ 3 2.02e+ 3 2.02e 3 2.01e+ 3 1.66e+ 0
```

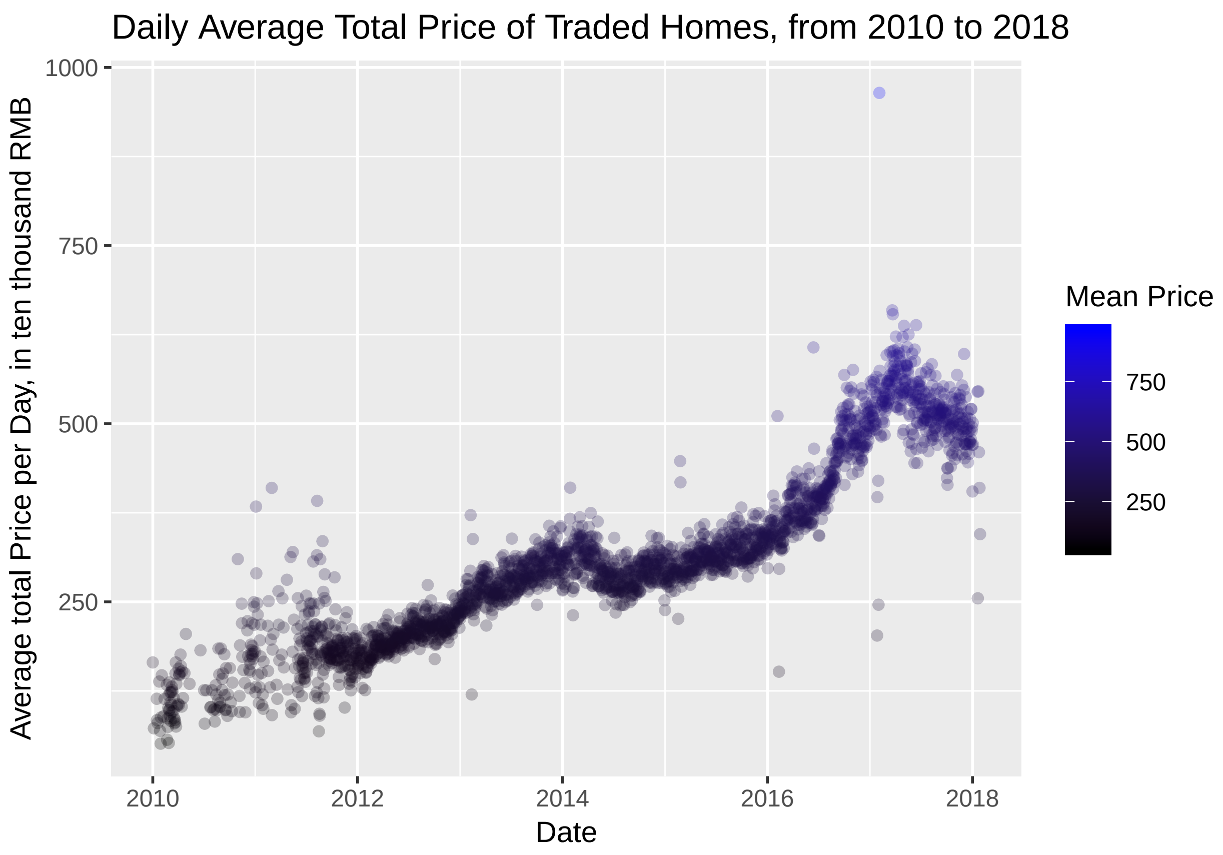
### 2.5 Plot average daily Price

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"))

data <- mutate(data, buldingAge = 2019 - constructionTime)

# 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) +
  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.

## 2.6 Implement Location Information

To run a regression with the location information, we need to think about how 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

To Calculate the distance of each object I use the haversine formula, which "determines the great-circle distance between two points on a sphere given their longitudes and latitudes." (source: https://en.wikipedia.org/wiki/Haversine_formula). Therefore I make use of the "geosphere" package.

```r
# Calculate the Distance
data <- data %>%
  mutate(bj_lat = 39.9042, bj_log = 116.4074) %>%
  mutate(dist = distHaversine(cbind(Lng, Lat), cbind(bj_log, bj_lat), r=6378137))
```

```
# Add District Factors
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')
  }
  else{return('missing')}
}
```

```r
data <- mutate(data, District = as.factor(sapply(district, makeDistrict)))
```

```r
# Download a Map of Beijing
# Longidude and Latidue Data are scratched from OpenStreetMap
map_bw <- openmap(c(39.6216, 115.9634000),
               c(40.2701000, 116.7778),
               minNumTiles=5,
               type = "stamen-toner")
plot(map_bw)

trans_by_district <- data %>% group_by(District) %>%
  summarise(Transactions = n(),
            AveragePrice = round(mean(totalPrice), digits = 2),
            AverageDistance = round(mean(dist), digits = 2))
trans_by_district
```

## 3 Analysis

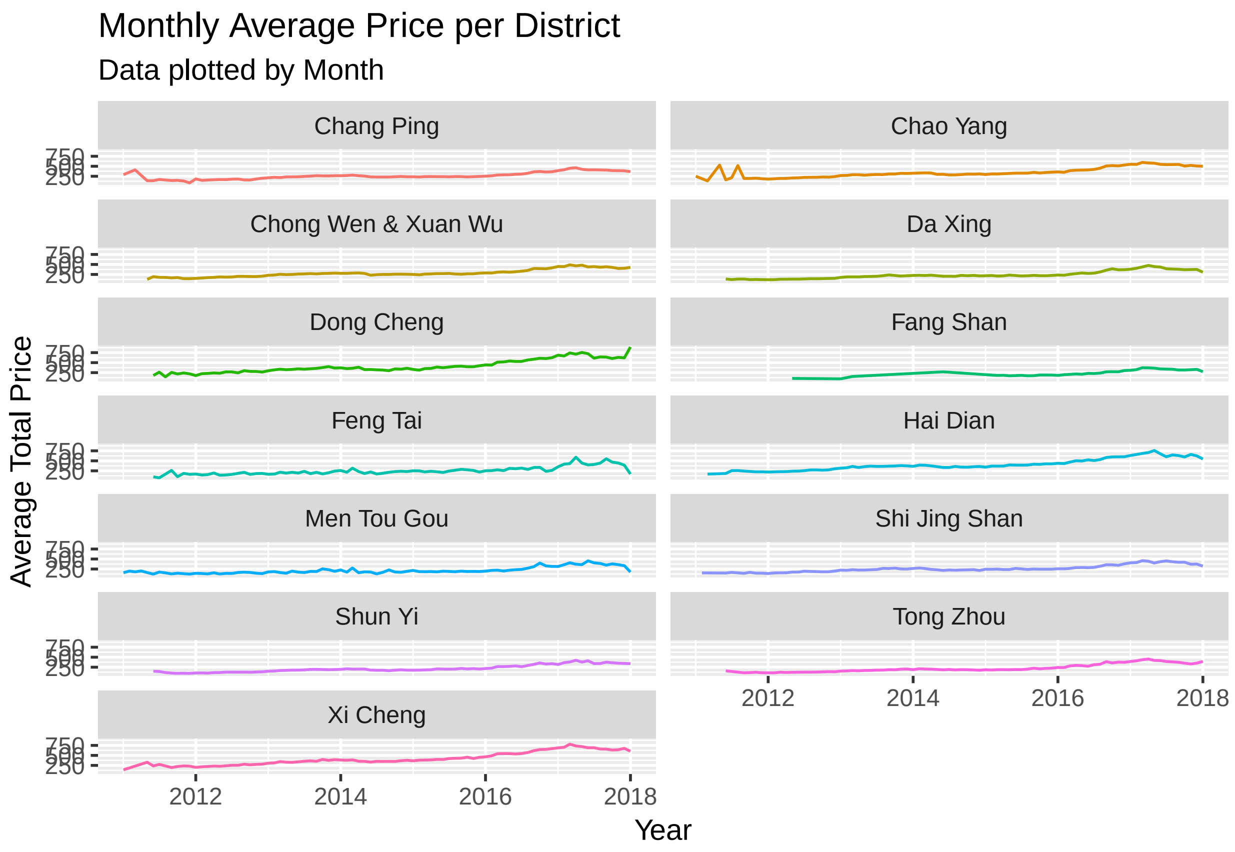
### 3.1 Plotting Transactions

```r
autoplot(OpenStreetMap::openproj(map_bw)) +
  geom_point(data = data, aes(x = Lng,
                              y = Lat,
                              shape = District,
                              colour = District)) +
  scale_shape_manual(values=1:nlevels(data$District)) +
  labs(title='Districts By Colour') +
  theme(axis.title.x = element_blank(),
       axis.text.x = element_blank(),
       axis.ticks.x = element_blank(),
       axis.title.y = element_blank(),
       axis.text.y = element_blank(),
       axis.ticks.y = element_blank())
# ggsave("map_sales_location_bw.png")
```

```r
autoplot(OpenStreetMap::openproj(map_bw)) +
  geom_point(data = data, aes(x = Lng,
                              y = Lat,
                              shape = District,
                              colour = price),
    size=1.3,alpha=.5) +
  scale_shape_manual(values=1:nlevels(data$District)) +
  scale_colour_gradient(low = "blue", high = "red") +
  labs(title='Houseprice per Sq.ft per District') +
  theme(axis.title.x = element_blank(),
       axis.text.x = element_blank(),
       axis.ticks.x = element_blank(),
       axis.title.y = element_blank(),
       axis.text.y = element_blank(),
       axis.ticks.y = element_blank())
# ggsave("map_houseprice_bw.png")
```

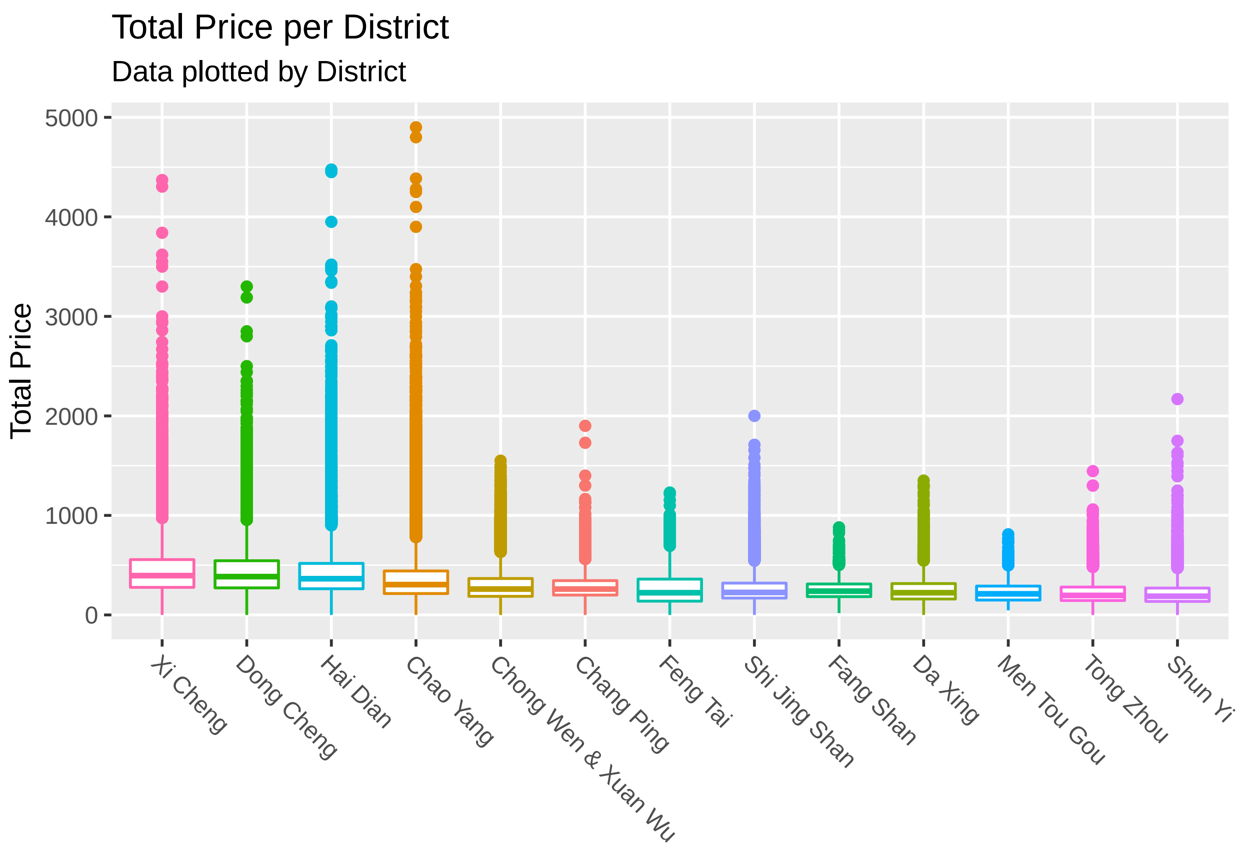
## 3.2 Price Evolution for each District by Year

```
# Monthly Average Price per District
data %>% 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")
```



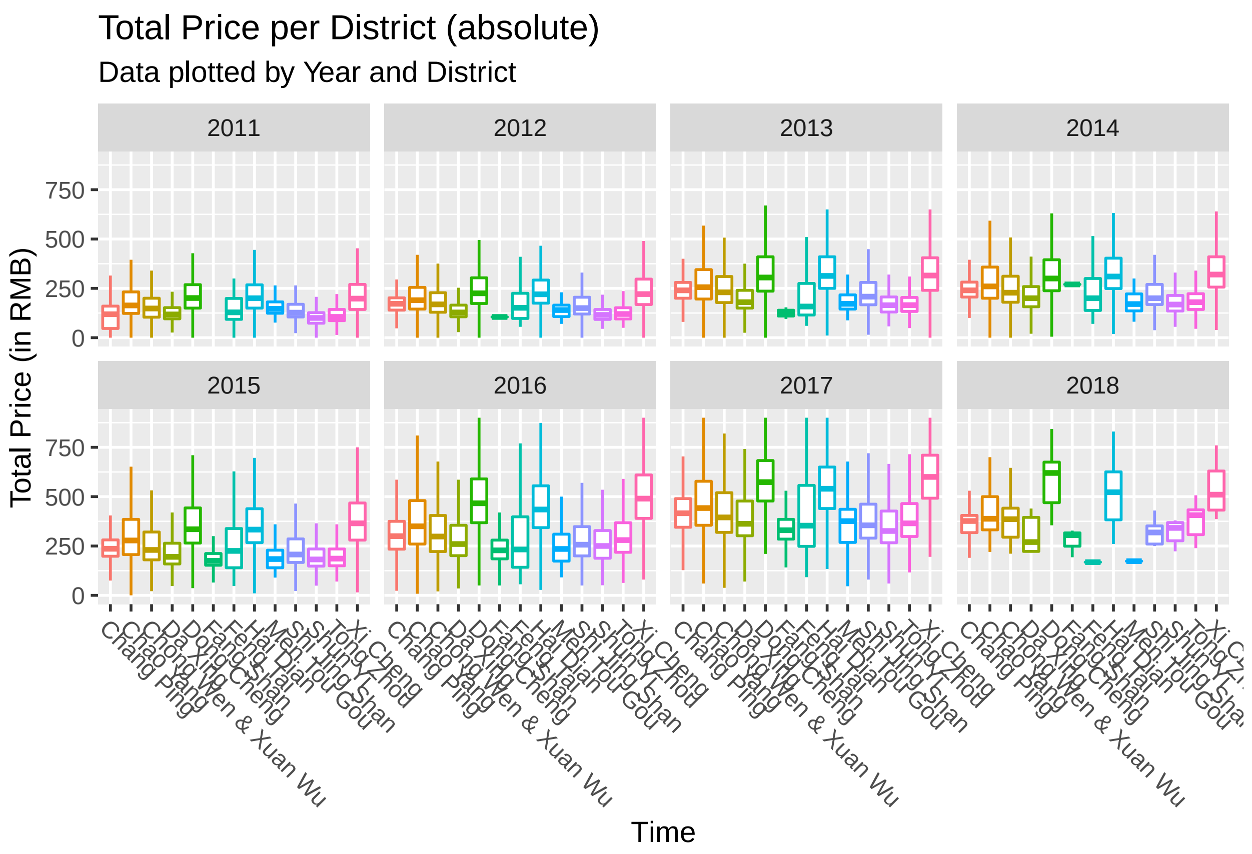
```
# ggsave("map_avgpriceperdistrict.png")
```

```
# Total Price Change per District (absolute)
data %>% filter(tradeYear > 2010) %>%
  group_by(tradeYear, District) %>%
  ggplot(aes(x = reorder(District, -totalPrice),
             y = totalPrice,
             color = District)) +
  geom_boxplot() +
  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",
       subtitle = "Data plotted by District",
       y = "Total Price")
```



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

```
# Monthly Price Change per District (absolute)
data %>% filter(tradeYear > 2010) %>%
  group_by(month=floor_date(tradeTime, "month"), District) %>%
  ggplot(aes(District, totalPrice, color = District)) +
  geom_boxplot(outlier.shape = NA) +
  scale_y_continuous(limits = c(0, 900)) +
  facet_wrap( ~ tradeYear   , ncol = 4) +
  labs(title = "Total Price per District (absolute)",
       subtitle = "Data plotted by Year and District",
       y = "Total Price (in RMB)",
       x = "Time") +
  theme(axis.text.x = element_text(angle = -45,
                                   vjust = 1,
                                   hjust = 0),
        legend.position="none")
```



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

### 3.3 Regressions

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.

#### 3.3.1 Linear Regression

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

ttest2 <- data %>%
  nest(-District) %>%
  mutate(fit = map(data, ~t.test(.$totalPrice)),
         p   = map_dbl(fit, "p.value"),
         results = map(fit, glance)) %>%
  unnest(results)

reg1.1 <- data %>%
  nest(-District) %>%
  mutate(fit = map(data, ~ lm(price ~
                                DOM +  livingRoom +  drawingRoom +  kitchen +
                                bathRoom + floor +  buildingType +  buldingAge +
                                buildingStructure + elevator +  fiveYearsProperty +
                                subway +  factor(tradeYear) +
                                dist + followers, data = .)),
         results = map(fit, glance)) %>%
  unnest(results) %>%
  ggplot(aes(x = factor(District), y = r.squared)) +
  geom_bar(stat = "identity") +
  labs(x = "District", y = expression(R^{2})) +
  theme(axis.text.x = element_text(angle = -45, vjust = 1, hjust = 0)) +
  ggsave("rsqrtperdistrict1.1.pdf")
```

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

```
reg1.2 <- data %>%
  nest(-District) %>%
  mutate(fit = map(data, ~ lm(totalPrice ~
                                DOM +  livingRoom +  drawingRoom +  kitchen +
                                bathRoom + floor +  buildingType +  buldingAge +
                                buildingStructure + elevator +  fiveYearsProperty +
                                subway +  factor(tradeYear) +
                                dist + followers, data = .)),
         results = map(fit, glance)) %>%
  unnest(results) %>%
  ggplot(aes(x = factor(District), y = r.squared)) +
  geom_bar(stat = "identity") +
  labs(x = "District", y = expression(R^{2})) +
  theme(axis.text.x = element_text(angle = -45, vjust = 1, hjust = 0)) +
  ggsave("rsqrtperdistrict1.2.pdf")
```

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

```
reg2.1 <- data %>%
  nest(-District) %>%
  mutate(fit = map(data, ~ lm(price ~
                                DOM +  livingRoom +  drawingRoom +  kitchen +
                                bathRoom + floor +  buildingType +  buldingAge +
                                buildingStructure + elevator +  fiveYearsProperty +
                                subway +  factor(tradeYear) +
                                dist + followers, data = .)),
         results2= map(fit, augment)) %>%
  unnest(results2) %>%
  ggplot(aes(x = price, y = .fitted, shape = District, colour = District)) +
  scale_shape_manual(values=1:nlevels(data$District)) +
  geom_abline(intercept = 0, slope = 1, alpha = .2) +  # Line of perfect fit
  geom_point() +
#  facet_grid(District ~ .) +
  theme_bw() +
  ggsave("modelfitted2.1.pdf")
```

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

```
reg2.2 <- data %>%
  nest(-District) %>%
  mutate(fit = map(data, ~ lm(totalPrice ~
                                DOM +  livingRoom +  drawingRoom +  kitchen +
                                bathRoom + floor +  buildingType +  buldingAge +
                                buildingStructure + elevator +  fiveYearsProperty +
                                subway +  factor(tradeYear) +
                                dist + followers, data = .)),
         results2= map(fit, augment)) %>%
  unnest(results2) %>%
  ggplot(aes(x = totalPrice, y = .fitted, shape = District, colour = District)) +
  scale_shape_manual(values=1:nlevels(data$District)) +
  geom_abline(intercept = 0, slope = 1, alpha = .2) +  # Line of perfect fit
  geom_point() +
#  facet_grid(District ~ .) +
  theme_bw() +
  ggsave("modelfitted2.2.pdf")
```

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

```
# Save Regression results into tibble
reg2tibble <- as_tibble(data %>%
      nest(-District) %>%
      mutate(fit = map(data, ~
                         lm(totalPrice ~
                              DOM +  livingRoom +  drawingRoom +  kitchen +
                              bathRoom + floor +  buildingType +  buldingAge +
                              buildingStructure + elevator +  fiveYearsProperty +
                              subway +  factor(tradeYear) +
                              dist + followers, data = .)),
         results2= map(fit, augment)) %>%
  unnest(results2))
```

```r
# Save each Regression plot as own file
plots <- reg2tibble %>%
  split(.$District) %>%
  map( ~ ggplot(., aes(x = .fitted, y = .resid)) +
         geom_point() +
         geom_abline(intercept = 0,
                     slope = 1,
                     alpha = .2))
paths <- stringr::str_c(names(plots), ".pdf")

pwalk(list(paths, plots), ggsave)
```

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

### 3.3.2 Logistic Regression

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

# Take only 75000 observations as my computing power is limited
ef_sample <- sample_n(data, size = 75000)

set.seed(123)
rF <- ef_sample %>%
  # Selecting data to work with
  na.omit() %>%
  select(totalPrice, District,
           DOM, livingRoom, drawingRoom, kitchen, bathRoom,
           floor, buildingType, buldingAge, buildingStructure,
           elevator, fiveYearsProperty, subway, tradeYear,
           dist, 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)

# 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")

# Plot each feature and its importance separated with each district
# to understand how each district is different in separate images.
plotrF <- rF %>%
  split(.$District) %>%
  map(~ggplot(., 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"))
paths <- stringr::str_c(names(plotrF), ".pdf")

pwalk(list(paths, plotrF), ggsave)
```

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

## 4 References

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