Biostatistics 625 Final Project Report

Group 2 members: Mukai Wang, Kangping Yang, Litian Zhou

December 20, 2019

Introduction

Housing costs in China have witnessed turbulent fluctuation in the past few years, especially in the country's capital Beijing. In this project, we design and implement an R shiny app that offers interactive data visualization and statistical insights about the housing price trend in Beijing. The data source is from Kaggle. It contains 300k+ transaction logs between 2011 and 2017.

Dataset Summary and Storage

The raw data contains 26 variables. After we remove columns that are not useful (e.g. ids) and columns with too many missing values, we retain ## variables that we are interested in. For details of the variables, please refer to our codebook in the github repo.

In order to better explain the association between price and time, we decide to generate a new variable called `season` based on the tradetime. The `season` variable sets the first season of 2010 as 1, the second season season of 2010 as 2, etc until the first season of 2018 as 33.

Because we need to subset the data in a variety of ways efficiently for the shiny app to display the data from various aspects, we decide to store the data in a relational database. Because we plan to deploy our shiny app online, an online database is our desired option. We ended up using Amazon Relational Database Service. Because our data is not too large, a free tier version is sufficient. The following example illustrates how we can query data from the database. For more details, please refer to our github repo.

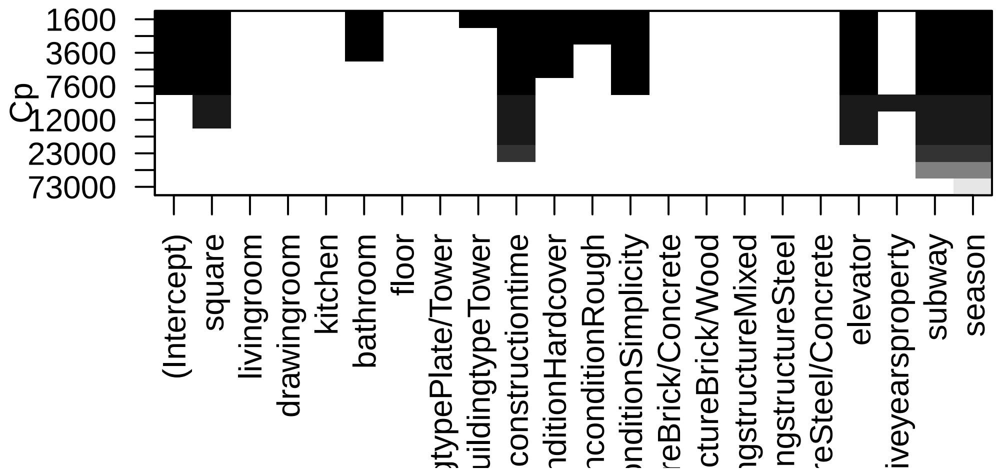
```r
# data_query_example.R gives us the handler to query data from the database
source("../data_cleaning_storage/data_query_example.R")
# gen_data is a function that acts as an easy api for
# querying data based on selected district and building type
# gen_full_data is a similar function that
# returns data with all covariates above to demonstrate full model.
exampledata = gen_full_data("all", c())
```

Statistical Model

We try to see if a linear model can capture the trend between price and other covariates. To decide which covariates need to be taken into account, we started from the full model. The full model contains all the covariates listed in the table above except the total price.

```r
library(leaps)
full_model <- lm(price ~ square + livingroom + drawingroom + kitchen + bathroom + floor +
                    buildingtype + constructiontime + renovationcondition +
                    buildingstructure + elevator + fiveyearsproperty +
                    subway + season, data = exampledata$data)
mat_full <- model.matrix(full_model)
selection <-regsubsets(mat_full,exampledata$data$price,int = F,really.big = T,
                        method = "exhaust", nvmax = 11)
```

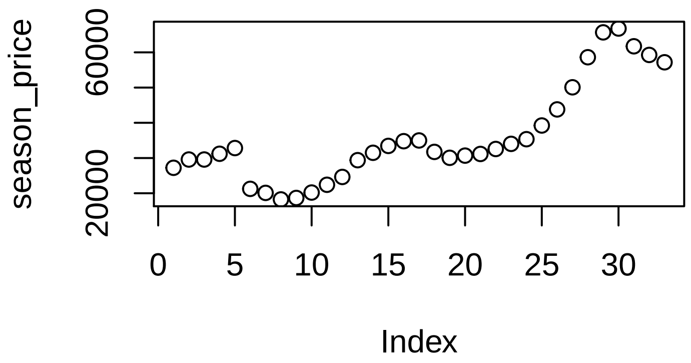
```
plot(selection,scale = "Cp")
```



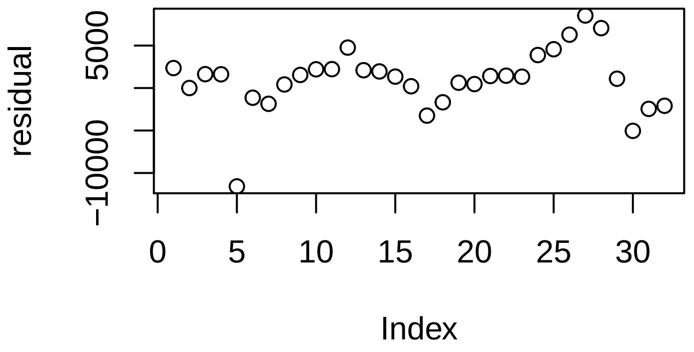
Based on the Mallow's Cp statistic, We choose covariates `square`,`bathroom`,`buildingtype`,`renovationcondition`, `district`, `elevator`, `subway` and `season`. To increase the model accuracy we decide to treat `season` as a factor.

```
exampledata$data$season <- as.factor(exampledata$data$season)
model_select <- lm(price ~ square + bathroom + buildingtype + constructiontime +
                     renovationcondition + elevator + subway + district +
                     season, data = exampledata$data)
season_price <- c(0,model_select$coefficients[23:54])
district_stat <- table(exampledata$data$district)
weight_dist <- district_stat/sum(district_stat)
reference <- model_select$coefficients[1] +
  mean(exampledata$data$bathroom)*model_select$coefficients[3] +
  sum(weight_dist*c(0,model_select$coefficients[12:22]))
```

```
season_price <- season_price + reference
##extract season coefficients as average selling price for time-series references.
plot(season_price) ## see the trend of housing price across seasons
```

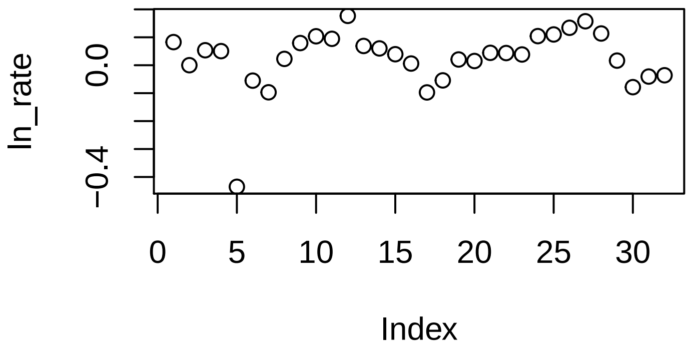


```
residual <- season_price[2:length(season_price)]-season_price[-length(season_price)]
plot(residual) ## residual plot between consecutive seasons
```

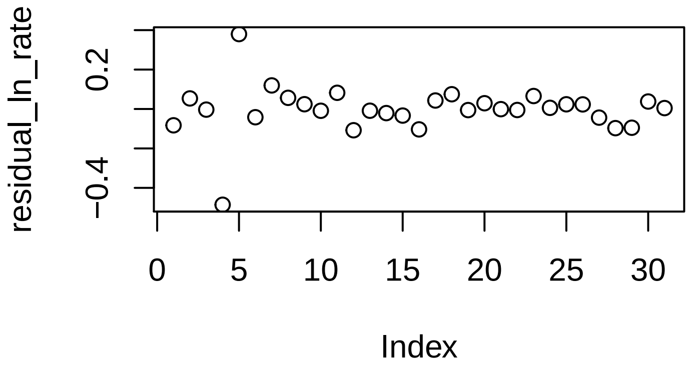


As we see price differences between adjacant seasons, we found strong correlation between these residuals and they tend to become unstable as the price went up. So we decided to calculate the log of the rate of the price change and observe its residual.

```
price_rate <- season_price[2:length(season_price)]/season_price[-length(season_price)]
ln_rate <- log(price_rate)
plot(ln_rate)
```



```
residual_ln_rate <- ln_rate[2:length(ln_rate)]-ln_rate[-length(ln_rate)]
plot(residual_ln_rate)
```



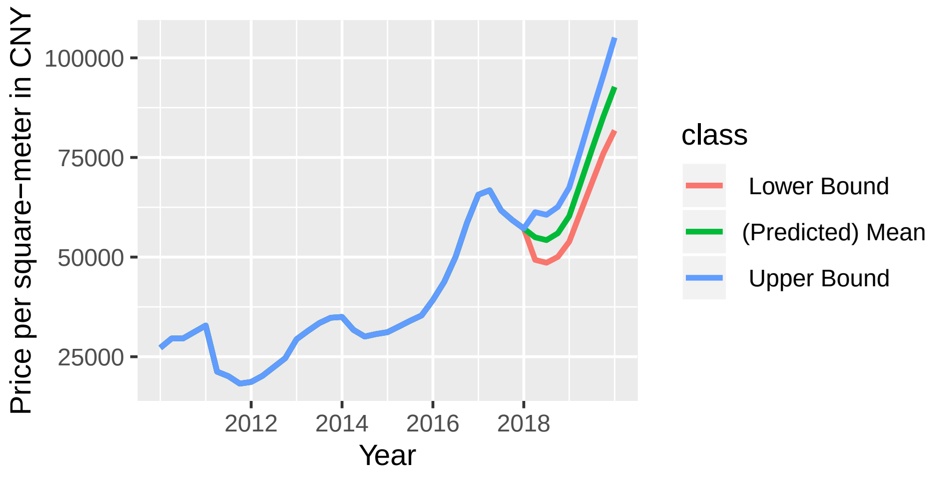
The residual plot looks better as the log rate change is more close to 0 and tend to become stable across the time. Therefore, we assume the log rate difference between adjacent seasons tend to be a constant around zero. For time series model ARIMA(p, d, q), this assumption indicates d = 1. As for p, we found the price tend to follow a similar pattern in about every two-year period. So we descide to choose $p = 8$ (8 seasons i.e. 2 years). As for q, we found the residual of log rate tend to be auto-regressive by 1 season. We descide to choose q = 1 The prediction model we use is ARIMA(8,1,1).

```
# the model fitting function is in another script
source("../Prediction_Model_Function.R")
# the function accepts the object returned from the gen_data function.
```

```
modeloutput = Filter_model(exampledata)
```

The predicted house price in 2018 to 2019 from the fitted ARIMA(8,1,1) is shown in the following price trend graph:

```
plots = ggplot(modeloutput$beta_data, aes(x = year, y = price , color = class)) +
      geom_line(size=1) + labs(x = 'Year' ,y = 'Price per square-meter in CNY') +
  scale_x_continuous(breaks = c(2012,2014,2016,2018))
plots
```



App Interface

We integrate data visualization and statistical inference into one webpage supported by Rshiny. The interface has four panels: map & markers panel, filter panel, plots panel and statistical inference panel.

Since we have more than thirty thousand house trading record, for the sake of reactivity, the map will random sampling seven hundred records which satisfies the filters.

In the left panel, user can apply filters to the markers shown on the map

Deliverable and Code Repository

Link to the online app.

Conclusion

In this project, We manage to set up an R shiny app that produces interactive visualizations and statistical inference for Beijing housing prices between 2010 and 2018.