King County Housing Prices

Jonathan McFadden

TCSS-551 : Big Data Analytics

Autumn 2017

Introduction

Overview

For our final project, we have chosen to analyze data covering housing sales in King County. To do this, we are using the data from the **Kaggle** *King County House Sales Prediction* page at

https://www.kaggle.com/harlfoxem/housesalesprediction

From this page, we sign-up for an account (*free, but required for downloading*) and then download the *zip* file containing the *CSV* file with the data.

Our goal is to use this data to create models for home sales in King County based on the feature information provided in the obtained data file. Our eventual goal is two-fold. First, we wish to create a model or models which will enable us to quantitatively predict house sale prices, using this data set as the basis for our model or models. Our other goal is to determine, based on the obtained data, which features are most important to the sale price of a house.

Data File

Our first task is to import, examine, and then give an overall description of the data. We are especially interested in the size and descriptive contents of the data file. Specifially, we want to know the number of sales contined within the data file and, especially, what parameters the data file uses to describe each house sale. Furthermore, we want to check the import to ensure that the data was initially complete, that it was then imported correctly, and that ***R*** is interpreting the imported data properly.

Import and First-Look

We begin by importing the data file into the '***houseDFo***( )' data frame. This data frame will serve as an intial data-frame, not the working one. This is because we may need an initial frame to reload as a we clean the data, allowing us to avoid having to reimport the CSV file over ang over again. Thus, we now import the CSV file into this initial data frame.

```
houseDFo <- read.csv("../houseData.csv")
```

We are now interested in the number of data-points contined within the data file. Thus, we want to see how many row ***R*** has imported.

```
nrow(houseDFo)
```

```
## [1] 21613
```

We also want to see how many descriptors the imported data uses to describe each house sale. Thus we want to see how many columns ***R*** has imported.

```
ncol(houseDFo)
```

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

In addition, we want to see what the labels for those columns are and what type of values the elements of each column have (*interger, numeric, string, etc.*)

```
sapply(houseDFo, class)
```

```
##            id          date         price      bedrooms     bathrooms
##     "numeric"      "factor"     "numeric"     "integer"     "numeric"
##   sqft_living      sqft_lot        floors    waterfront          view
##     "integer"     "integer"     "numeric"     "integer"     "integer"
##     condition         grade    sqft_above sqft_basement      yr_built
##     "integer"     "integer"     "integer"     "integer"     "integer"
##  yr_renovated       zipcode           lat          long sqft_living15
##     "integer"     "integer"     "numeric"     "numeric"     "integer"
##    sqft_lot15
##     "integer"
```

From above, it is clear that the **date** column did not import as a *date*, instead importing as a *factor*. Therefore, we will now examine the first few rows of the imported data to see what may have caused the issues with imporation.

```
head(houseDFo)
```

```
##           id            date   price bedrooms bathrooms sqft_living sqft_lot
## 1 7129300520 20141013T000000  221900        3      1.00        1180     5650
## 2 6414100192 20141209T000000  538000        3      2.25        2570     7242
## 3 5631500400 20150225T000000  180000        2      1.00         770    10000
## 4 2487200875 20141209T000000  604000        4      3.00        1960     5000
## 5 1954400510 20150218T000000  510000        3      2.00        1680     8080
## 6 7237550310 20140512T000000 1225000        4      4.50        5420   101930
##   floors waterfront view condition grade sqft_above sqft_basement yr_built
## 1      1          0    0         3     7       1180             0     1955
## 2      2          0    0         3     7       2170           400     1951
## 3      1          0    0         3     6        770             0     1933
## 4      1          0    0         5     7       1050           910     1965
## 5      1          0    0         3     8       1680             0     1987
## 6      1          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

Clean the Data

Missing Data

First, we will check to see if there are any missing data points.

```
houseDFo[!complete.cases(houseDFo),]
```

```
##  [1] id            date          price         bedrooms      bathrooms
##  [6] sqft_living   sqft_lot      floors        waterfront    view
```

```
## [11] condition     grade         sqft_above    sqft_basement yr_built
## [16] yr_renovated  zipcode       lat           long          sqft_living15
## [21] sqft_lot15
## <0 rows> (or 0-length row.names)
```

Since there are no missing data points, we can move on to the dates.

Dates

From the first few rows of the data table seen above, it is clear that we must first strip the *"T000000"* string at the end of every date. To do this, we require the **stringr** library. Thus, we import **stringr**

```
library(stringr)
```

so we can now strip the offending substrings. Before stripping these substrings, we create a copy of our initial data frame, ***houseDFo***( ), so that our initial import data frame will remain untouched, and therefore available for reloading other data frames. Thus, we create the copy and strip the substrings, storing the result in the copied data frame ***houseDFo1***( ).

```
houseDFo1 <- houseDFo
houseDFo1$date = str_replace(houseDFo$date, "T000000", "")
```

We now examine the result of this

```
head(houseDFo1)
```

```
##           id     date   price bedrooms bathrooms sqft_living sqft_lot floors
## 1 7129300520 20141013  221900        3      1.00        1180     5650      1
## 2 6414100192 20141209  538000        3      2.25        2570     7242      2
## 3 5631500400 20150225  180000        2      1.00         770    10000      1
## 4 2487200875 20141209  604000        4      3.00        1960     5000      1
## 5 1954400510 20150218  510000        3      2.00        1680     8080      1
## 6 7237550310 20140512 1225000        4      4.50        5420   101930      1
##   waterfront view condition grade sqft_above sqft_basement yr_built
## 1          0    0         3     7       1180             0     1955
## 2          0    0         3     7       2170           400     1951
## 3          0    0         3     6        770             0     1933
## 4          0    0         5     7       1050           910     1965
## 5          0    0         3     8       1680             0     1987
## 6          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

The dates are now just strings of numbers with the format '*yyyymmdd*'; therefore, we can use the date conversion method from ***R*** to convert these dates.

```
houseDFo1 <- transform(houseDFo1, date = as.Date(date, "%Y%m%d"))
```

To ensure that the conversion to dates happend properly, we will no check the column data types followed by looking at the first few rows of the data.

```
sapply(houseDFo1, class)
```

```
##           id         date        price     bedrooms    bathrooms
```

```
##     "numeric"        "Date"     "numeric"     "integer"     "numeric"
##   sqft_living      sqft_lot        floors    waterfront          view
##     "integer"     "integer"     "numeric"     "integer"     "integer"
##     condition         grade    sqft_above sqft_basement      yr_built
##     "integer"     "integer"     "integer"     "integer"     "integer"
##  yr_renovated       zipcode           lat          long sqft_living15
##     "integer"     "integer"     "numeric"     "numeric"     "integer"
##    sqft_lot15
##     "integer"
```

```
head(houseDFo1)
```

```
##           id       date   price bedrooms bathrooms sqft_living sqft_lot floors
## 1 7129300520 2014-10-13  221900        3      1.00        1180     5650      1
## 2 6414100192 2014-12-09  538000        3      2.25        2570     7242      2
## 3 5631500400 2015-02-25  180000        2      1.00         770    10000      1
## 4 2487200875 2014-12-09  604000        4      3.00        1960     5000      1
## 5 1954400510 2015-02-18  510000        3      2.00        1680     8080      1
## 6 7237550310 2014-05-12 1225000        4      4.50        5420   101930      1
##   waterfront view condition grade sqft_above sqft_basement yr_built
## 1          0    0         3     7       1180             0     1955
## 2          0    0         3     7       2170           400     1951
## 3          0    0         3     6        770             0     1933
## 4          0    0         5     7       1050           910     1965
## 5          0    0         3     8       1680             0     1987
## 6          0    0         3    11       3890          1530     2001
##   yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1            0   98178 47.5112 -122.257          1340       5650
## 2         1991   98125 47.7210 -122.319          1690       7639
## 3            0   98028 47.7379 -122.233          2720       8062
## 4            0   98136 47.5208 -122.393          1360       5000
## 5            0   98074 47.6168 -122.045          1800       7503
## 6            0   98053 47.6561 -122.005          4760     101930
```

Since the results for the date conversions are as desired, we can now store the data in a final data frame followed by moving on to begining our analysis.

```
houseDF <- houseDFo1
```

We will also create a version of the data with the **ID** column stripped out.

```
houseDFa <- houseDF[-c(1)]
```

Analysis

Initial Analysis

To begin our analysis, we will look at the basic statistics of every column (*except the date*).

```
sapply(houseDFa[-c(1)], function(x) list(mean=mean(x),
                                         stdev=sd(x, na.rm=TRUE)))
```

```
##       price    bedrooms  bathrooms sqft_living sqft_lot floors    waterfront
## mean  540088.1 3.370842  2.114757  2079.9      15106.97 1.494309  0.007541757
## stdev 367127.2 0.9300618 0.7701632 918.4409    41420.51 0.5399889 0.0865172
##       view      condition grade    sqft_above sqft_basement yr_built
## mean  0.2343034 3.40943   7.656873 1788.391   291.509       1971.005
```

```
## stdev 0.7663176 0.650743  1.175459 828.091    442.575       29.37341
##       yr_renovated zipcode  lat       long      sqft_living15 sqft_lot15
## mean  84.40226     98077.94 47.56005  -122.2139 1986.552      12768.46
## stdev 401.6792     53.50503 0.1385637 0.1408283 685.3913      27304.18
```

```
library(nnet)
modelA <- lm(price ~., data=houseDFa)
summary(modelA)
```

```
##
## Call:
## lm(formula = price ~ ., data = houseDFa)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1306672   -98900    -8963    77327  4330103
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.618e+06  2.933e+06   1.574  0.11539
## date           1.165e+02  1.213e+01   9.608  < 2e-16 ***
## bedrooms      -3.588e+04  1.888e+03 -19.005  < 2e-16 ***
## bathrooms      4.137e+04  3.247e+03  12.741  < 2e-16 ***
## sqft_living    1.502e+02  4.376e+00  34.327  < 2e-16 ***
## sqft_lot       1.257e-01  4.782e-02   2.629  0.00858 **
## floors         7.158e+03  3.589e+03   1.995  0.04610 *
## waterfront     5.826e+05  1.732e+04  33.628  < 2e-16 ***
## view           5.260e+04  2.136e+03  24.629  < 2e-16 ***
## condition      2.774e+04  2.351e+03  11.799  < 2e-16 ***
## grade          9.624e+04  2.149e+03  44.791  < 2e-16 ***
## sqft_above     3.084e+01  4.351e+00   7.088 1.40e-12 ***
## sqft_basement         NA         NA      NA       NA
## yr_built      -2.618e+03  7.251e+01 -36.113  < 2e-16 ***
## yr_renovated   2.079e+01  3.649e+00   5.698 1.23e-08 ***
## zipcode       -5.807e+02  3.292e+01 -17.643  < 2e-16 ***
## lat            6.053e+05  1.072e+04  56.487  < 2e-16 ***
## long          -2.136e+05  1.311e+04 -16.300  < 2e-16 ***
## sqft_living15  2.195e+01  3.441e+00   6.381 1.79e-10 ***
## sqft_lot15    -3.825e-01  7.311e-02  -5.232 1.69e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200800 on 21594 degrees of freedom
## Multiple R-squared:  0.701,  Adjusted R-squared:  0.7008
## F-statistic:  2813 on 18 and 21594 DF,  p-value: < 2.2e-16
```