Shiny Application for Exploring Boston Airbnb Data

Tayler Li

12/05/2021

Preprocessing Data

The dataset on which this shiny app is based is `listings.csv`. I conducted several preprocessing steps on it before exploring the data inside.

1. In Excel, I selected a few columns to keep in the dataset while deleting all others. The names of the kept columns are shown below. Note that all variables related to customer review are deleted because there are too many missing values and attempts to predict review ratings based on other variables and fill in the missed spots introduce collinearity.

```
df <- read.csv('listings.csv')
dim(df)
```

```
## [1] 3585   30
```

```
colnames(df)
```

```
##  [1] "host_response_time"        "host_response_rate"
##  [3] "host_acceptance_rate"      "host_is_superhost"
##  [5] "host_total_listings_count" "host_verifications"
##  [7] "host_has_profile_pic"      "host_identity_verified"
##  [9] "neighbourhood_cleansed"    "latitude"
## [11] "longitude"                 "is_location_exact"
## [13] "property_type"             "room_type"
## [15] "accommodates"              "bathrooms"
## [17] "bedrooms"                  "beds"
## [19] "bed_type"                  "amenities"
## [21] "price"                     "security_deposit"
## [23] "cleaning_fee"              "minimum_nights"
## [25] "availability_30"           "availability_60"
## [27] "availability_90"           "availability_365"
## [29] "instant_bookable"          "cancellation_policy"
```

2. I filtered on the `host_response_time` variable and keeps only four levels: “within an hour”, “within a few hours”, “within a day”, “a few days or more”. All the other entries in this column, whether blank or not, are considered invalid and the entire row in which these entries are are deleted.

```
dropind <- which(!(df$host_response_time %in% c("within an hour","within a few hours","within a day","a few days or more")))    # drop index
df <- df[-dropind,]
df <- df %>% mutate(host_response_time=as.factor(host_response_time))
dim(df)
```

```
## [1] 3085   30
```

3. I filtered out rows that contain empty values for the variable `property_type`.

```
dropind <- which(df$property_type=="")
df <- df[-dropind,]
dim(df)
```

```
## [1] 3083   30
```

4. For binary variables in the dataset, I replace the entries with more informative ones.

```
df <- df %>%
  mutate(host_is_superhost=as.factor(ifelse(host_is_superhost=="t","Superhost","Not a Superhost")),
         host_has_profile_pic=as.factor(ifelse(host_has_profile_pic=="t","Yes","No")),
         host_identity_verified=as.factor(ifelse(host_identity_verified=="t","Verfied","Not Verified")),
         is_location_exact=as.factor(ifelse(is_location_exact=="t","Exact Location","Location Not Accurat
e")),
         instant_bookable=as.factor(ifelse(instant_bookable=="t","Instantly Bookable","Not Instantly Booka
ble")))
```

5. I fill all the blank entries in the dataset with value 0.

```
df[df==""] <- 0
```

6. Special characters like $ and % are removed.

```
df <- df %>%
  mutate(price=str_replace_all(price,",",""),
         security_deposit=str_replace_all(security_deposit,",",""),
         cleaning_fee=str_replace_all(cleaning_fee,",","")) %>%
  mutate(host_response_rate=as.numeric(str_replace(host_response_rate,"\\%",""))/100,
         host_acceptance_rate=as.numeric(str_replace(host_acceptance_rate,"\\%",""))/100,
         price=as.numeric(str_replace(price,"\\$","")),
         security_deposit=as.numeric(str_replace(security_deposit,"\\$","")),
         cleaning_fee=as.numeric(str_replace(cleaning_fee,"\\$","")))
```

7. Amenities listed in the `amenities` column are turned into integer values counting the total number of amenities listed. Similar opreations are conducted on the `host_verifications` column.

```
amenity <- df %>%
  summarize(amenities=lengths(str_split(amenities,",")))
host_verification <- df %>%
  summarize(host_verifications=lengths(str_split(host_verifications,",")))
df <- df %>% mutate(host_verifications=host_verification$host_verifications, amenities=amenity$amenities)
```

8. Entries of some variables are turned into numeric values or factors or subject to further calculation or adjustment.

```
df <- df %>%
  mutate(host_total_listings_count=as.numeric(host_total_listings_count),
         accommodates=as.numeric(accommodates),
         bathrooms=as.numeric(bathrooms),
         bedrooms=as.numeric(bedrooms),
         beds=as.numeric(beds),
         availability_30=as.numeric(availability_30),
         availability_60=as.numeric(availability_60),
         availability_90=as.numeric(availability_90),
         availability_365=as.numeric(availability_365),
         cancellation_policy=case_when(cancellation_policy=="flexible"~"Flexible",
                                       cancellation_policy=="moderate"~"Moderate",
                                       cancellation_policy=="strict"~"Strict",
                                       cancellation_policy=="super_strict_30"~"Super Strict"),
         minimum_nights=as.numeric(minimum_nights),
         neighbourhood_cleansed=as.factor(neighbourhood_cleansed),
         longitude=as.numeric(longitude),
         latitude=as.numeric(latitude),
         `Price Level`=case_when(price<=85~"$~85 per day",
                                 price>85 & price<=150~"$85~150 per day",
                                 price>150 & price<=220~"$150~220 per day",
                                 price>220~"$220~ per day")) %>%
  mutate(`Price Level`=as.factor(`Price Level`))
```

The processed data is stored in `base.Rdata`.

```
save(df,file="base.Rdata")
```

App Design

This Shiny App consists of 5 tabs: “Overview”, “Host”, “Property”, “Policies”, and “Location”. The “Overview” tab contains a histogram showing the distribution of listing prices across the entire dataset. Users can adjust the bin width and the price range of the histogram.

In the other 4 tabs, I plotted the one-to-one relationship between listing price and variables related to the name of each tab. For categorical variables, I employed boxplots to depict the distribution of price at each level of the variable. For continuous variables, I made scatterplots with smoothing to visualize their relationship with price; users can select the smoothing method (linear model, GAM, or LOWESS) incorporated in the scatterplot. One special feature of the “Location” tab is that it has a map containing points that represent each of the listings in the dataset, and the color of the point indicates the pricing level; users can select to view listings in all neighborhoods or in one certain neighborhood.

For details of the Shiny App, please refer to `app.R`.