Airbnb in New York City

Carole Mattmann und Jonas Zuercher

13 Februar 2020

Packages used:

```
library(dplyr)
library(tidyverse)
library(geosphere)
library(ggplot2)
library(tinytex)
library(leaflet)
```

Summary

We are exploring a dataset of airbnb listings in New York City in 2019.

Analyses on the prices of the listings were run and models were created to predict prices of the listings.The best model to calculate the price that was found includes 5 variables:

- room type (factor with 3 levels, entire appartment being the highest and shared room the lowest)
- distance to timessquare (negative effect)
- availability (positive effect)
- neighbourhood group (factor with 5 levels, Manhattan being the highest and Bronx the lowest)
- minimum nights (negative effect)

The airbnb dataset was merged with a dataset concerning incidents (e.g. crimes) in the concerning neigbourhoods.

The airbnb dataset is vizualized in two maps in the last chapter in two ways:

- All datapoints are visualized on the map and coloured depending on their price range.
- The most expensive and cheapest housings are visualized with personalized markers. Also an distance calculator is included in the map.

Data import and cleaning

Airbnb dataset

The dataset was downloaded from: https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data

Import

```
AB_NYC <- read.csv("../01_data/AB_NYC_2019.csv", header=TRUE)
```

Overview of dataset

```
str(AB_NYC,width=80,strict.width="cut")
```

```
## 'data.frame':    48895 obs. of  16 variables:
##  $ id                            : int  2539 2595 3647 3831 5022 5099 5121 517..
##  $ name                          : Factor w/ 47906 levels "","'Fan'tastic",..:..
##  $ host_id                       : int  2787 2845 4632 4869 7192 7322 7356 896..
##  $ host_name                     : Factor w/ 11453 levels "","'Cil","-TheQuee"..
##  $ neighbourhood_group           : Factor w/ 5 levels "Bronx","Brooklyn",..: 2..
##  $ neighbourhood                 : Factor w/ 221 levels "Allerton","Arden Hei"..
##  $ latitude                      : num  40.6 40.8 40.8 40.7 40.8 ...
##  $ longitude                     : num  -74 -74 -73.9 -74 -73.9 ...
##  $ room_type                     : Factor w/ 3 levels "Entire home/apt",..: 2 ..
##  $ price                         : int  149 225 150 89 80 200 60 79 79 150 ...
##  $ minimum_nights                : int  1 1 3 1 10 3 45 2 2 1 ...
##  $ number_of_reviews             : int  9 45 0 270 9 74 49 430 118 160 ...
##  $ last_review                   : Factor w/ 1765 levels "","2011-03-28",..: 1..
##  $ reviews_per_month             : num  0.21 0.38 NA 4.64 0.1 0.59 0.4 3.47 0...
##  $ calculated_host_listings_count: int  6 2 1 1 1 1 1 1 1 4 ...
##  $ availability_365              : int  365 355 365 194 0 129 0 220 0 188 ...
```

Following changes are made to the dataset:

Remove price 0

Remove all listings with price 0

```
AB_NYC <-AB_NYC[AB_NYC$price > 0,]
```

Add log price

Add logarithmic price for analysis purposes

```
AB_NYC <- cbind(AB_NYC,price_log = log(AB_NYC$price))
```

Remove inactive listings

Remove inactive listings and make new dataset

```
AB_NYC_available <- AB_NYC %>%
  filter(availability_365 > 0)
```

Add distance to Times Square to model

We want to make a statement about how central the place is. Therefore the distance to Times Square is caculated using the latitude and longitude of the listings. The package "geosphere" is used.

Times Square, Manhattan, NY, USA, Latitude and longitude coordinates are: 40.758896, -73.98513

```r
coord <- cbind(AB_NYC_available$longitude,AB_NYC_available$latitude)
dist.timessquare <- distGeo(p1=coord, p2=c(-73.985130, 40.758896))
AB_NYC_available <- cbind(AB_NYC_available,dist.timessquare)
```

Prepare dataset for merging with the second dataset

We want to merge this dataset with the Incident dataset. Therefore we need to transform the neighbour_group column by changing the letters to lowercase, delete empty spaces and defining the entries as factors.

```r
# write neighbourhood group entries in lower case
AB_NYC_available$neighbourhood_group<-tolower(AB_NYC_available$neighbourhood_group)

#remove spaces from neighbourhood groups
AB_NYC_available$neighbourhood_group <-gsub(" ","", AB_NYC_available$neighbourhood_group)

# neighbourhood group as factor
AB_NYC_available$neighbourhood_group<-factor(AB_NYC_available$neighbourhood_group)
```

Incidents dataset

The dataset was downloaded from: https://data.cityofnewyork.us/City-Government/Agency-Performance-Mapping-Indicator
gsj6-6rwm

```r
Ind_NYC<- read.csv("../01_data/Indicators_NYC.csv")
```

Overview of dataset

```r
str(Ind_NYC)
```

```
## 'data.frame':    3659 obs. of  13 variables:
##  $ Agency              : Factor w/ 189 levels "DCA","DEP","DOB",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Geographic.Unit     : Factor w/ 5 levels "","Community District",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Geographic.Identifier: Factor w/ 156 levels "","1","10","100",..: 156 154 153 143 142 141 140 139
##  $ Indicator           : Factor w/ 62 levels "","Air complaints received",..: 46 46 46 46 46 46 46
##  $ FY2011              : Factor w/ 1699 levels "","< 5","<5",..: 1019 1045 1407 370 1288 201 828 11
##  $ FY2012              : Factor w/ 1703 levels "","< 5","<5",..: 934 1175 1161 606 831 267 1007 104
##  $ FY2013              : Factor w/ 1839 levels "","< 5","<5",..: 1229 1259 733 1661 577 157 883 124
##  $ FY2014              : Factor w/ 1791 levels "","< 5","<5",..: 935 1069 1322 1506 1040 155 1275 1
##  $ FY2015              : Factor w/ 1940 levels "","< 5","<5",..: 996 794 1101 1553 1124 252 978 123
##  $ FY2016              : Factor w/ 1868 levels "","< 5","<5",..: 839 1369 1391 148 1349 224 1082 12
##  $ FY2017              : Factor w/ 1935 levels "","< 5","<5",..: 630 658 1140 340 965 1760 1093 981
##  $ FY2018              : Factor w/ 1784 levels "","< 5","<5",..: 700 623 683 576 1014 1193 855 1024
##  $ FY2019              : num  14 26 34 25 40 66 25 51 47 18 ...
```

Producer a dataframe for 2019 including Neighbourhoods, indicators and the incident numbers

First a dataframe which only contains the neighbour_group, the indicator and the incidents of 2019 has been produced.

```
Ind_NYC_2019_cleaned<-data.frame("neighbourhood_group"= Ind_NYC$Geographic.Identifier, "Indicator"=Ind_
head(Ind_NYC_2019_cleaned)
```

```
##   neighbourhood_group                     Indicator Incidents
## 1     Staten Island 3 Resolved Consumer Complaints        14
## 2     Staten Island 2 Resolved Consumer Complaints        26
## 3     Staten Island 1 Resolved Consumer Complaints        34
## 4           Queens 14 Resolved Consumer Complaints        25
## 5           Queens 13 Resolved Consumer Complaints        40
## 6           Queens 12 Resolved Consumer Complaints        66
```

Transform dataframe for further processing

We want to merge this dataframe with the airbnb dataset. Therefore we need to transform the neighbour_group column by removing the numbers of the neighbourhood_group entries, changing the letters to lowercase, delete empty spaces and defining the entries as factors.

```
#Remove numbers from neighbourhood_group column
Ind_NYC_2019_cleaned$neighbourhood_group <-gsub("[0-9]","", Ind_NYC_2019_cleaned$neighbourhood_group)
#Remove empty spaces from neighbourhood_group column
Ind_NYC_2019_cleaned$neighbourhood_group <-gsub(" ","", Ind_NYC_2019_cleaned$neighbourhood_group)
#Change all lettes from neighbourhood_group column to lower case
Ind_NYC_2019_cleaned$neighbourhood_group<-tolower(Ind_NYC_2019_cleaned$neighbourhood_group)
#Change type of the neighbourhood_group column to factor
Ind_NYC_2019_cleaned$neighbourhood_group<-factor(Ind_NYC_2019_cleaned$neighbourhood_group)

#overview
head(Ind_NYC_2019_cleaned)
```

```
##   neighbourhood_group                     Indicator Incidents
## 1        statenisland Resolved Consumer Complaints        14
## 2        statenisland Resolved Consumer Complaints        26
## 3        statenisland Resolved Consumer Complaints        34
## 4              queens Resolved Consumer Complaints        25
## 5              queens Resolved Consumer Complaints        40
## 6              queens Resolved Consumer Complaints        66
```

Summarise Observation per neighbourhood_group

We want only one row per neighbourhood_group entry and Indicator. Therefore we first group the entries by neighbourhood_groups and then summarize the incidents numbers. Because some entries are given in percentage those entries result in NA entries after this process and need to be removed with the filter feature.

```
# sum of incidents per neighbourhood group and indicator
Summary_Ind_NYC_2019<-Ind_NYC_2019_cleaned %>%
  group_by(neighbourhood_group=Ind_NYC_2019_cleaned$neighbourhood_group,Indicator) %>%
  summarise(Observations=sum(Incidents,na.rm = TRUE))

# Remove entries without a neighbourhood group from the dataframe
Summary_Ind_NYC_2019<-filter(Summary_Ind_NYC_2019,neighbourhood_group != "")
head(Summary_Ind_NYC_2019)
```

```
## # A tibble: 6 x 3
## # Groups:   neighbourhood_group [1]
##   neighbourhood_gro~ Indicator                                  Observations
##   <fct>              <fct>                                             <dbl>
## 1 bronx              Air complaints received                          536
## 2 bronx              Asbestos complaints received                     212
## 3 bronx              Average response time to life-threatenin~          7.44
## 4 bronx              Average response time to life-threatenin~          5.13
## 5 bronx              Average response time to structural fires          4.36
## 6 bronx              Citywide acceptability rating for the cl~       1137.
```

Transform dataframe into a nested dataframe

The dataframe is transformed into a nested dataframe to improve readiablity after merging it with the airbnb dataset.

```
# nested indicators
NYC_nest<-Summary_Ind_NYC_2019 %>%
  nest(Indicator=c(Indicator, Observations))
head(NYC_nest)
```

```
## # A tibble: 5 x 2
## # Groups:   neighbourhood_group [6]
##   neighbourhood_group      Indicator
##   <fct>                <list<df[,2]>>
## 1 bronx                      [38 x 2]
## 2 brooklyn                   [38 x 2]
## 3 manhattan                  [37 x 2]
## 4 queens                     [38 x 2]
## 5 statenisland               [38 x 2]
```

Merge datasets

The processed dataset are merged together by the neighbour_group column.

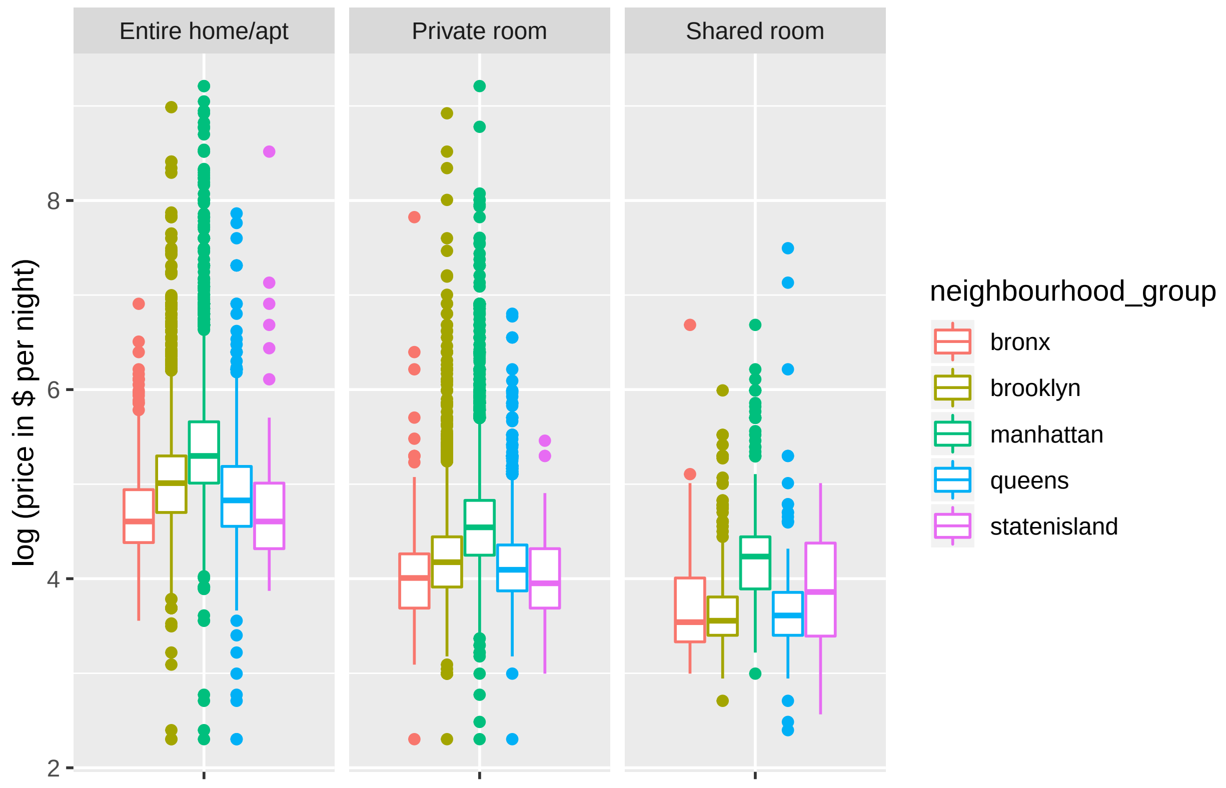
```
#Join both datasets
NYC<-left_join(AB_NYC_available,NYC_nest, by="neighbourhood_group")

# neighbourhood group as factor
NYC$neighbourhood_group<-factor(NYC$neighbourhood_group)

# show new dataframe
head(NYC,1)
```

```
##     id                             name host_id host_name
## 1 2539 Clean & quiet apt home by the park    2787      John
##   neighbourhood_group neighbourhood latitude longitude    room_type price
## 1            brooklyn    Kensington 40.64749 -73.97237 Private room   149
##   minimum_nights number_of_reviews last_review reviews_per_month
## 1              1                 9  2018-10-19              0.21
##   calculated_host_listings_count availability_365 price_log
## 1                              6              365  5.003946
##   dist.timessquare
## 1         12418.34
##
## 1 2.0000, 3.0000, 7.0000, 8.0000, 9.0000, 12.0000, 13.0000, 14.0000, 18.0000, 19.0000, 20.0000, 21.0
```

Data visualisation

Distribution of prices by room types and neighbourhood

```r
ggplot(data = AB_NYC_available,
       mapping = aes(y = price_log,
                     x = "",
                     group = neighbourhood_group,
                     colour = neighbourhood_group)) +
  geom_boxplot() +
  facet_wrap(. ~ room_type)+
  xlab("")+
  ylab("log (price in $ per night)")
```

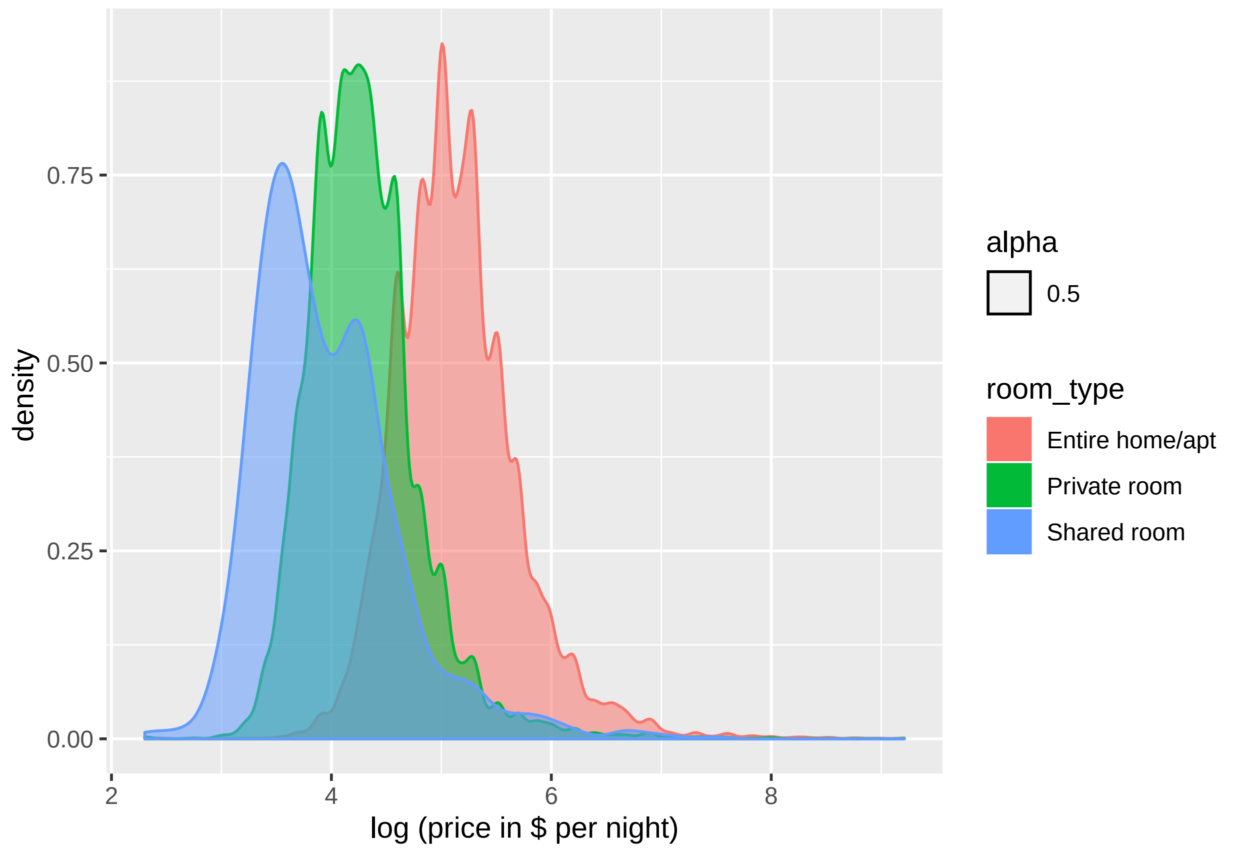


Prices of the room type “entire home/apt” have the highest median, followed by “private room”" and lastly “shared room”“, which is not surprising. 25. and 75. quantile for”entire home/apt" and “private room” are similarly distributed, for shared room there is no clear pattern.

For all room types, median prices in neighbourhood “Manhattan” are the highest. For for “entire home/apt” and “private room” the second highest mediam prices are in Manhattan.

Distribution of prices

```
ggplot(data = AB_NYC,
       mapping = aes(x = price_log,
                     group = room_type,
                     colour = room_type,
                     fill = room_type,
                     alpha = 0.5)) +
  geom_density() +
  xlab("log (price in $ per night)")+
  ylab("density ")
```

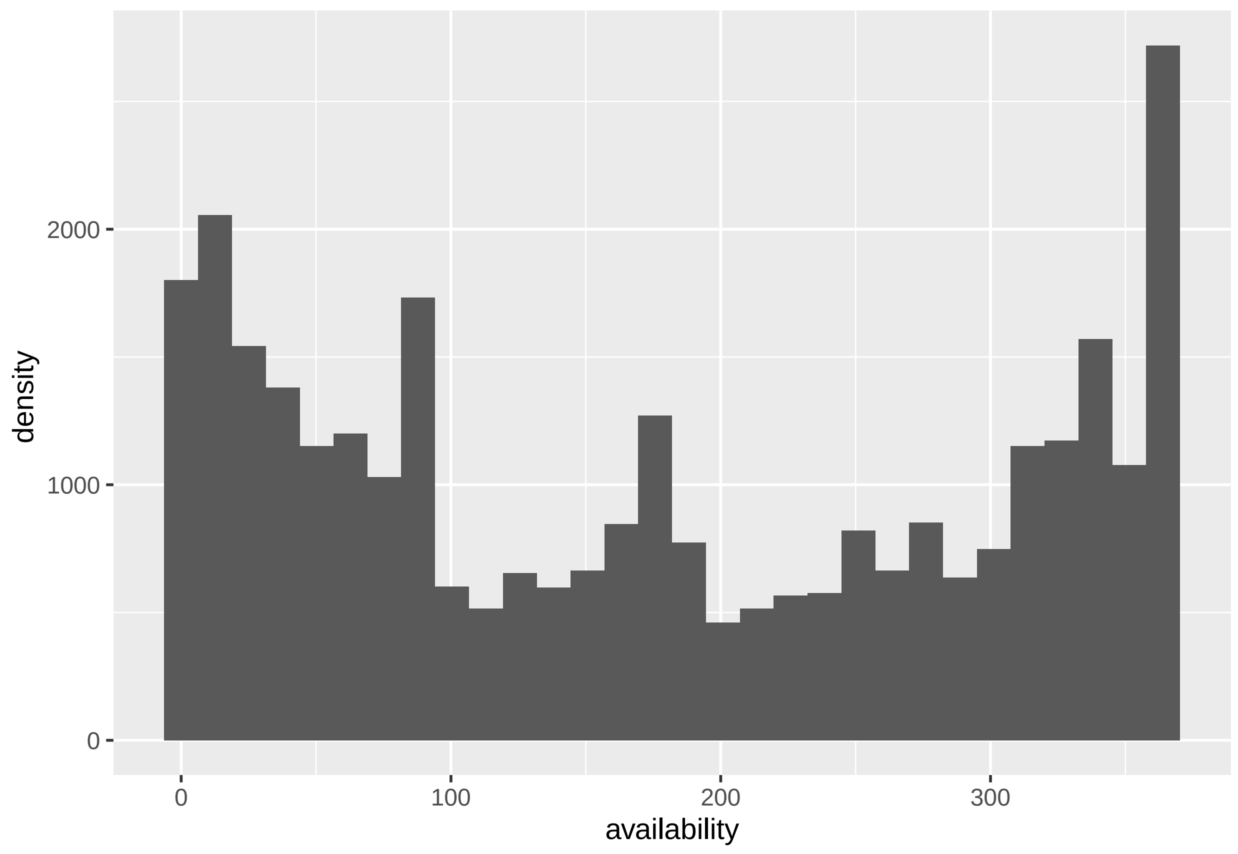


Prices for all room types are skewed to the right. Even with logarithmic display of prices, this is still clearly the case.

Availability of appartments

```
ggplot(data = AB_NYC_available,
       mapping = aes(x = availability_365)) +
  geom_histogram() +
  xlab("availability")+
  ylab("density ")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There are a lot of listings with very low or very high (almost year round) availablity. Listings with no available days in 2019 were removed from the dataset. This distribution was not taken into account when looking at the prices.

Possible models to calculate the price of an airbnb

Simple linear models

Impact of several variables on the price are analysed. The highest R2 is reached with "room type".

```
##simple linear models

# price ~ neighbourhood group

lm.hood <- lm (data=AB_NYC_available, price_log~neighbourhood_group)
summary(lm.hood)
```

```
##
## Call:
## lm(formula = price_log ~ neighbourhood_group, data = AB_NYC_available)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7663 -0.4698 -0.0473  0.3886  4.3652
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        4.25517    0.02199 193.516  < 2e-16 ***
## neighbourhood_groupbrooklyn        0.36688    0.02279  16.096  < 2e-16 ***
## neighbourhood_groupmanhattan       0.81367    0.02272  35.818  < 2e-16 ***
## neighbourhood_groupqueens          0.12670    0.02421   5.233 1.68e-07 ***
## neighbourhood_groupstatenisland    0.10551    0.04263   2.475   0.0133 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6644 on 31349 degrees of freedom
## Multiple R-squared:  0.1491, Adjusted R-squared:  0.1489
## F-statistic:  1373 on 4 and 31349 DF,  p-value: < 2.2e-16
```

```
# price ~ room type

lm.type <- lm (data=AB_NYC_available, price_log~room_type)
summary(lm.type)
```

```
##
## Call:
## lm(formula = price_log ~ room_type, data = AB_NYC_available)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8872 -0.3695 -0.0658  0.2816  4.8867
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.189793   0.004377 1185.79   <2e-16 ***
## room_typePrivate room -0.866270   0.006468 -133.92   <2e-16 ***
## room_typeShared room  -1.280409   0.019660  -65.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5627 on 31351 degrees of freedom
## Multiple R-squared:  0.3895, Adjusted R-squared:  0.3895
## F-statistic: 1e+04 on 2 and 31351 DF,  p-value: < 2.2e-16
```

```
# price ~ dist.timessquare

lm.dist <- lm (data=AB_NYC_available, price_log~dist.timessquare)
summary(lm.dist)
```

```
##
```

```
## Call:
## lm(formula = price_log ~ dist.timessquare, data = AB_NYC_available)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8052 -0.4752 -0.0408  0.3890  4.4443
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.211e+00  6.900e-03  755.27   <2e-16 ***
## dist.timessquare -5.913e-05  7.753e-07  -76.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6615 on 31352 degrees of freedom
## Multiple R-squared:  0.1565, Adjusted R-squared:  0.1565
## F-statistic:  5816 on 1 and 31352 DF,  p-value: < 2.2e-16
```

Multiple linear model

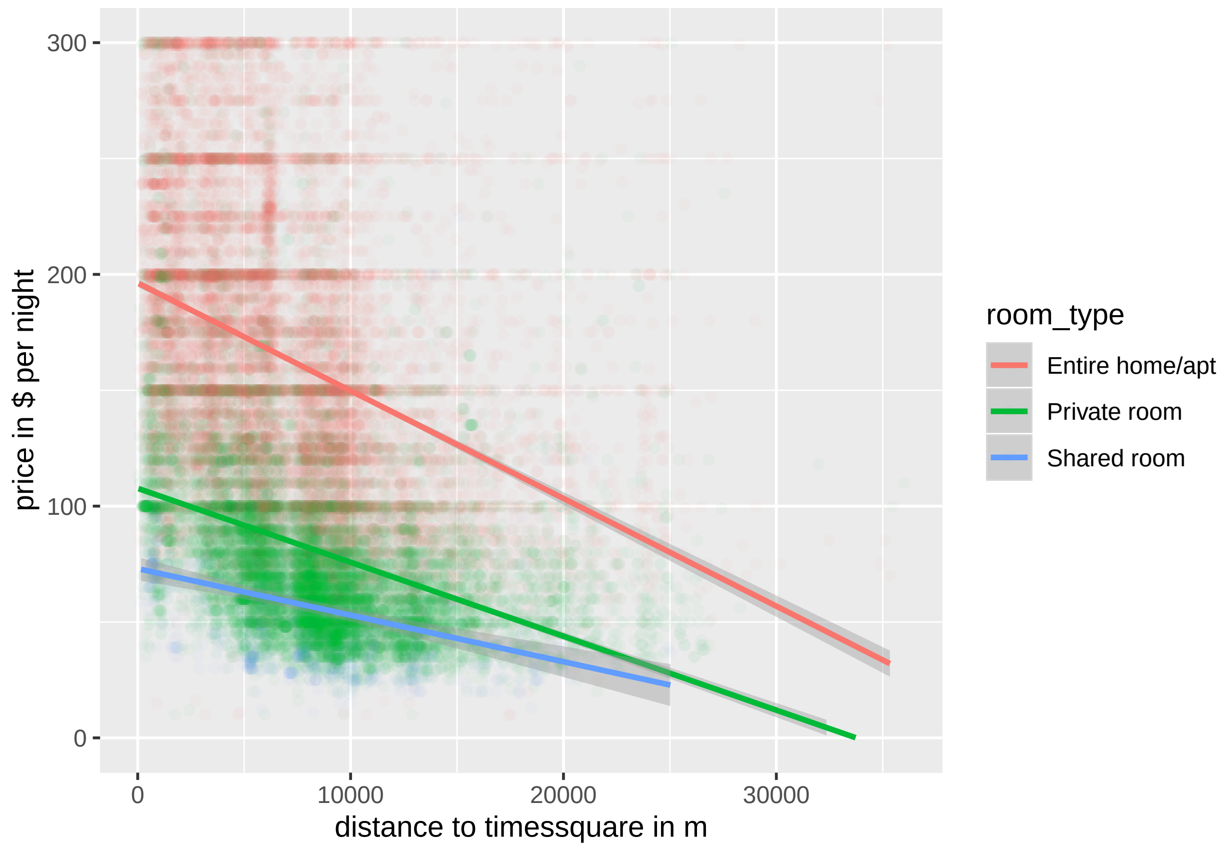
A linear model using "distance to timessquare" and "room type" (as factor) with interactions to calculate the price is applied.

```r
#distance and room type on price (with interaction)
lm.dist.type.interact <- lm (data=AB_NYC_available, price_log~dist.timessquare*room_type)
summary(lm.dist.type.interact)
```

```
##
## Call:
## lm(formula = price_log ~ dist.timessquare * room_type, data = AB_NYC_available)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0389 -0.3348 -0.0639  0.2365  4.7750
##
## Coefficients:
##                                         Estimate Std. Error t value
## (Intercept)                            5.480e+00  6.991e-03 783.825
## dist.timessquare                      -4.357e-05  8.550e-07 -50.955
## room_typePrivate room                 -7.825e-01  1.148e-02 -68.163
## room_typeShared room                  -1.214e+00  3.403e-02 -35.682
## dist.timessquare:room_typePrivate room -7.460e-07  1.274e-06  -0.586
## dist.timessquare:room_typeShared room  -2.747e-07  3.568e-06  -0.077
##                                       Pr(>|t|)
## (Intercept)                             <2e-16 ***
## dist.timessquare                        <2e-16 ***
## room_typePrivate room                   <2e-16 ***
## room_typeShared room                    <2e-16 ***
## dist.timessquare:room_typePrivate room   0.558
## dist.timessquare:room_typeShared room    0.939
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.5229 on 31348 degrees of freedom
## Multiple R-squared:  0.4729, Adjusted R-squared:  0.4728
## F-statistic:  5625 on 5 and 31348 DF,  p-value: < 2.2e-16
```

```r
ggplot(data = AB_NYC_available,
       mapping = aes(y = price,
                     x = dist.timessquare,
                     colour = room_type,
                     group = room_type)) +
  geom_point(alpha = 0.03) +
  xlab("distance to timessquare in m")+
  ylab("price in $ per night")+
  ylim(0,300)+
  geom_smooth(method="lm")
```



There is a tendency for all room types that the price is lower if the place is further from Times Square. The interactions are not significant.

Multiple linear model by choosing smallest RSS

A multiple linear model is created. The best model to calculate the price we can find includes 5 variables:

- room type (factor with 3 levels, entire appartment being the highest and shared room the lowest)
- distance to timessquare (negative effect)
- availability (positive effect)

- neighbourhood group (factor with 5 levels, Manhattan being the highest and Bronx the lowest)
- minimum nights (negative effect)

```
#full model
lm.full <- lm (data=AB_NYC_available, price_log~room_type
               +neighbourhood_group
               +minimum_nights
               +number_of_reviews
               +calculated_host_listings_count
               +availability_365
               +dist.timessquare)

#empty model
lm.empty <- lm (data=AB_NYC_available, price_log~NULL)
add1(lm.empty,scope=lm.full)

#choose value with smallest RSS
lm.1 <- update(lm.empty,.~.+room_type)
add1(lm.1,scope=lm.full)
lm.2 <- update(lm.1,.~.+dist.timessquare)
add1(lm.2,scope=lm.full)
lm.3 <- update(lm.2,.~.+availability_365)
add1(lm.3,scope=lm.full)
lm.4 <- update(lm.3,.~.+neighbourhood_group)
add1(lm.4,scope=lm.full)
lm.5 <- update(lm.4,.~.+minimum_nights)
add1(lm.5,scope=lm.full)
```

```
summary(lm.5)
```

```
##
## Call:
## lm(formula = price_log ~ room_type + dist.timessquare + availability_365 +
##     neighbourhood_group + minimum_nights, data = AB_NYC_available)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0328 -0.3241 -0.0652  0.2332  4.8822
##
## Coefficients:
##                                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     5.082e+00  2.077e-02  244.710  < 2e-16
## room_typePrivate room          -7.823e-01  5.995e-03 -130.499  < 2e-16
## room_typeShared room           -1.235e+00  1.785e-02  -69.173  < 2e-16
## dist.timessquare               -3.073e-05  8.321e-07  -36.926  < 2e-16
## availability_365                6.388e-04  2.311e-05   27.645  < 2e-16
## neighbourhood_groupbrooklyn     1.415e-01  1.779e-02    7.956 1.83e-15
## neighbourhood_groupmanhattan    3.210e-01  1.908e-02   16.821  < 2e-16
## neighbourhood_groupqueens       4.895e-02  1.863e-02    2.627  0.00861
## neighbourhood_groupstatenisland 1.754e-01  3.306e-02    5.304 1.14e-07
## minimum_nights                 -1.930e-03  1.226e-04  -15.741  < 2e-16
##
## (Intercept)                    ***
## room_typePrivate room           ***
## room_typeShared room            ***
## dist.timessquare                ***
## availability_365                ***
## neighbourhood_groupbrooklyn     ***
## neighbourhood_groupmanhattan    ***
## neighbourhood_groupqueens       **
## neighbourhood_groupstatenisland ***
## minimum_nights                  ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5089 on 31344 degrees of freedom
## Multiple R-squared:  0.5009, Adjusted R-squared:  0.5008
## F-statistic:  3496 on 9 and 31344 DF,  p-value: < 2.2e-16
```

Interactive maps with the leaflet package

In this part the leaflet package is used to produced interactive maps of the merged dataset.

Interactive map with three coloured price groups

The first interactive map shows all housings with a colored marker. The color varies with the price per night for the housings.

Prepare groups for visualization in the map

To adjust the color depending on the price of the datapoints a if-else loop is used. The same approach is used to classify the datapoint into the three groups Cheap, Medium and Expensive.

```r
# Function that chose the color depending on the price of the housing
getColor <- function(NYC) {
  sapply(NYC$price, function(price) {
    if(price <= 50) {
      "green"
    } else if(price <= 200) {
      "orange"
    } else {
      "red"
    } })
}

# Function that chose a group name depending on the price of the housing
getGroup <- function(NYC) {
  sapply(NYC$price, function(price) {
    if(price <= 50) {
      "Cheap"
    } else if(price <= 200) {
      "Medium"
    } else {
      "Expensive"
    } })
}
```

Visualization of a color coded interactive map

Its possible to zoom and change settings in this interactive map. Two different base map layers can be chosen. Furthermore only certain price groups like Cheap can be shown in the map. The green points are housings up to 50 dollar per night. The yellow points are housings up to 200 $ per night and the red ones are the housing that are even more expensive. Each marker shows detailed information with a popup if its clicked.

```
map<-leaflet(NYC) %>%
  #Base layer
  addTiles(group = "OSM") %>%
  addProviderTiles(providers$CartoDB.Positron,group="GREY")%>%

  # Overlay groups - Adding price dependant colored housing markers
  addCircleMarkers(lng = NYC$longitude,
                   lat = NYC$latitude,
                   color=getColor(NYC),
                   popup = paste0("Price per night: ", as.character(NYC$price), " $",
                                  "<br>","Room type: ",NYC$room_type,
                                  "<br>","Distance to Times Square: ",as.character(round(NYC$dis
                   radius=0.1,group = getGroup(NYC)) %>%

  #LayercOntrol
  addLayersControl(baseGroups = c("OSM","GREY"),
                   overlayGroups = c("Cheap","Medium","Expensive"))


map
```

Interactive map with the most expensive and cheapest housings

The second interactive map shows the most expensive and the cheapest housings of the dataset with personalized markers.

Prepare dataframe for cheapest and most expensive spots in New York

First the most expensive and the cheapest housings have to be found in the dataset and saved in a dataframe.

Prepare Icons

The icons have to be prepared for the visualization. Therefore the pictures are imported.

```
castle<-makeIcon(
  iconUrl = "../01_data/castle.png",
  iconWidth = 20, iconHeight = 20,
)
```

```
hovel<-makeIcon(
  iconUrl = "../01_data/hovel.png",
  iconWidth = 20, iconHeight = 20,
)
```

Visualization of the most expensive and cheapest airbnb housings in New York

Its possible to zoom and change settings in this interactive map. Two different base map layers can be chosen. Furthermore only the cheapest or the most expensive datapoints can be shown in the map. The most expensive housing is visualized with a custom castle icon. The cheapest housings are visualized with a custom hovel icon. Each marker shows detailed information with a popup if its clicked.

```
map2<-leaflet(NYC) %>%
  #Base
  addTiles(group = "OSM") %>%
  addProviderTiles(providers$CartoDB.Positron,group="GREY")%>%

  # Measure

  addMeasure(primaryLengthUnit = "kilometers",activeColor="red")%>%

  # Overlay groups
  addMarkers(lng=df_exp$longitude,
             lat=df_exp$latitude,
              icon= castle, popup = paste0("Price per night: ", as.character(df_cheap$exp), " $",
                                        "<br>","Room type: ",df_exp$room_type,
                                        "<br>","Distance to Times Square: ",as.character(round(df_exp$
             group=("Most expensive Airbnb Housing")) %>%
  addMarkers(lng=df_cheap$longitude,
             lat=df_cheap$latitude,
             icon= hovel, popup = paste0("Price per night: ", as.character(df_cheap$price), " $",
                                        "<br>","Room type: ",df_cheap$room_type,
                                        "<br>","Distance to Times Square: ",as.character(round(df_cheap
             group=("Cheapest Airbnb Housing")) %>%
  #LayerCOntrol
  addLayersControl(baseGroups = c("OSM","GREY"), overlayGroups = c("Most expensive Airbnb Housing","Chea

map2
```