Modelling Midterm Project

Jacob Burke

16/11/2019

AirBNB Data - Predicting Listing Prices in Europe

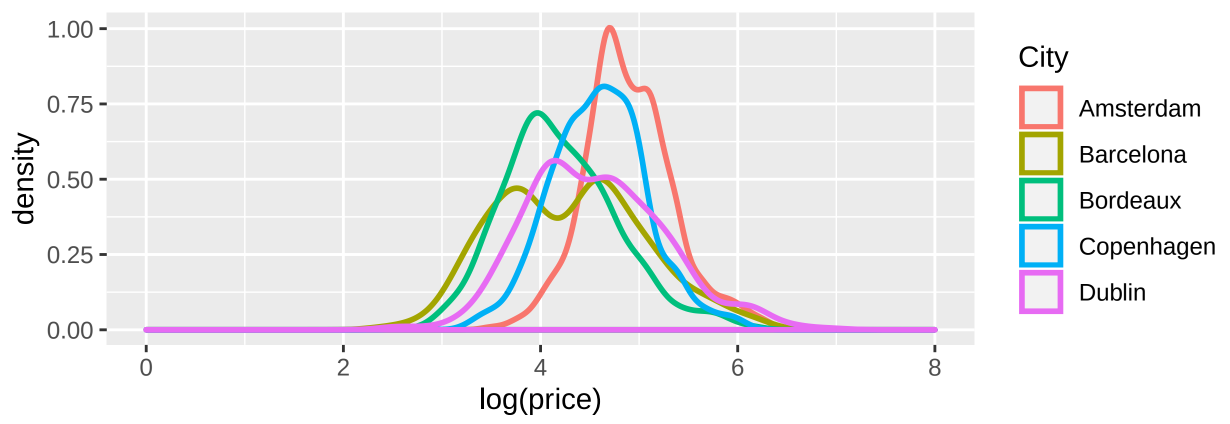
The data source that we'll be using for this report consists of AirBnB listings from 10 different cities within Europe (Amsterdam, Barcelona, Bordeaux, Dublin, Copenhagen, London, Madrid, Nice, Paris, and Rome). The goal for this analysis, is to look to see if we can make predictions on prices of AirBnB listings, based on the European cities that the listing are found along with additional attributes.

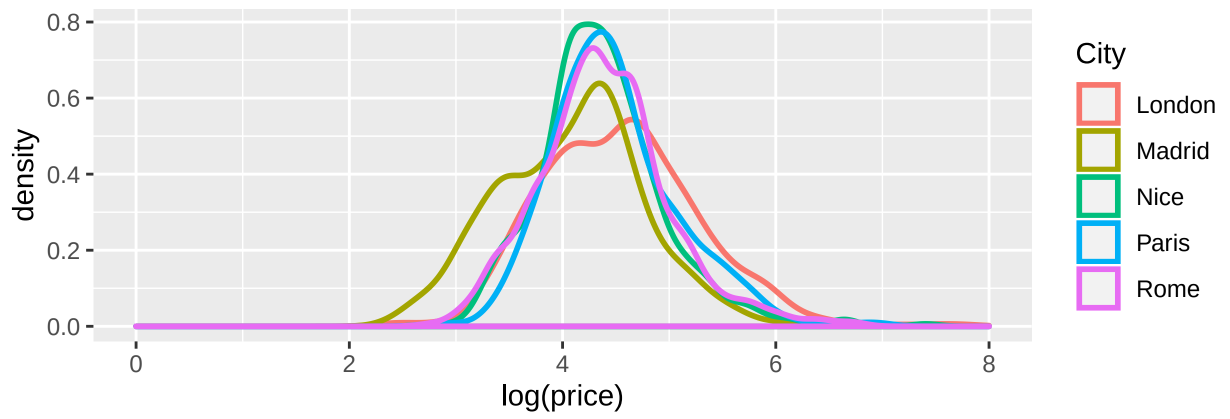
First off, the AirBNB data was cleaned and combined for analysis. Info on how this was performed can be found in the Appendix. Once put together this is a preview of the data set that we will be using for this analysis.

```
##   X  room_id   host_id       room_type reviews overall_satisfaction
## 1 1  7005410  34197767 Entire home/apt       3                  4.0
## 2 2  7854067   6870134    Private room     109                  5.0
## 3 3 17279326  68362369 Entire home/apt       0                  0.0
## 4 4 16235663 106117460 Entire home/apt       5                  4.5
## 5 5  6905082  25676251 Entire home/apt       0                  0.0
## 6 6  3017539   4389536 Entire home/apt       1                  0.0
##   accommodates bedrooms price      City
## 1            4        2   250 Amsterdam
## 2            2        0   131 Amsterdam
## 3            4        2   156 Amsterdam
## 4            3        0   128 Amsterdam
## 5            2        1   203 Amsterdam
## 6            4        2   200 Amsterdam
```

EDA Part 1

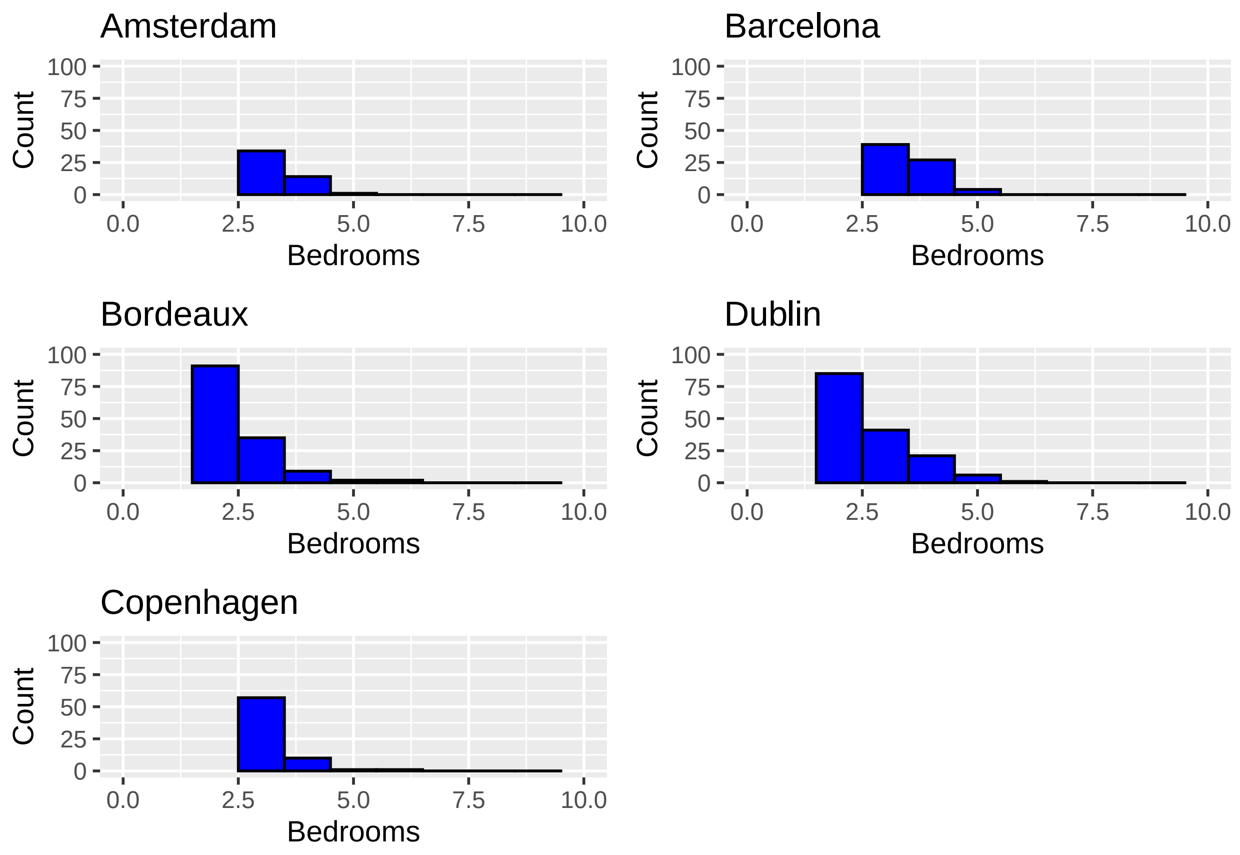
Now that we have the data cleaned and combined into one data set, grouped amongst cities, we can now begin EDA. Given that for this analysis we are looking to make predictions on the income of prices, we should first look at the overall distributions of price for each city. It is important to note, that in the data collection description "price" variable refers to the cost of stay at a listing for one night in US dollars.

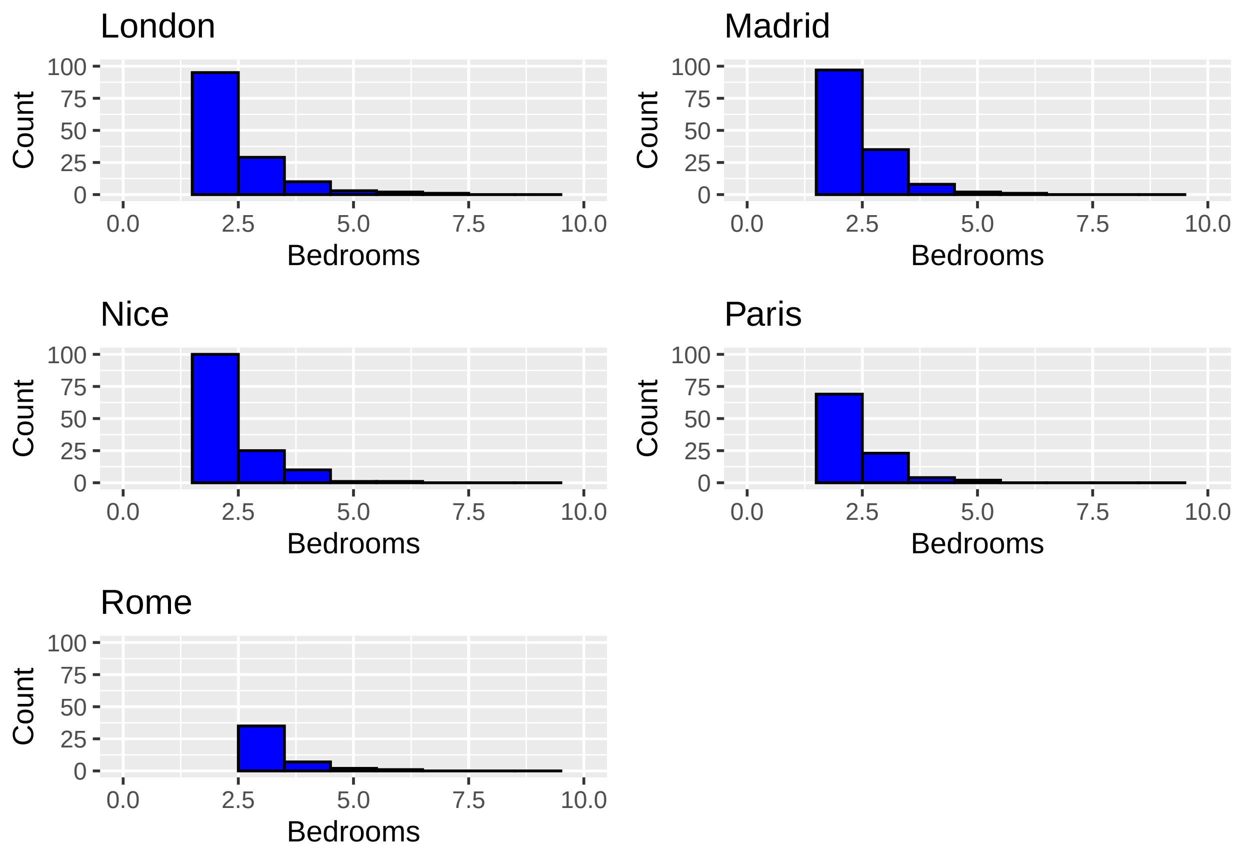




For the purpose of centering the data, we can plot the price distributions at a log base 10 scale. Here we can see that the majority of prices within all cities fall within the range exp(3.5) and exp(6) (ie. \$33 - \$403). London, Madrid, and Dublin have the widest ranges of prices overall. Copenhagen and Amsterdam look to have the most expensive average listing prices.

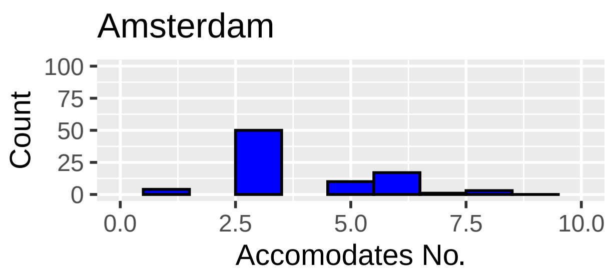
Now, we want to further explore the predictors that intuitively should all have effects on listing prices (bedrooms, accomadations, reviews, overall satisfaction, and room type).

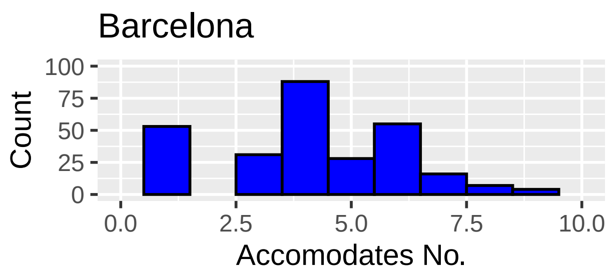

Amsterdam
Barcelona
Bordeaux
Dublin
Copenhagen
Count
Bedrooms
0
25
50
75
100
0.0
2.5
5.0
7.5
10.0

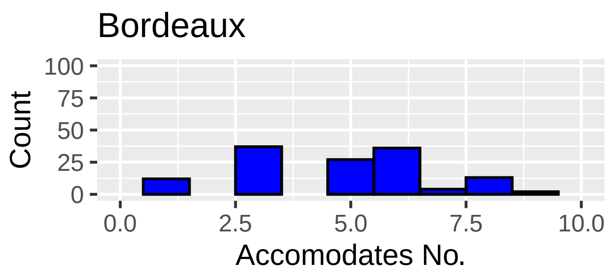


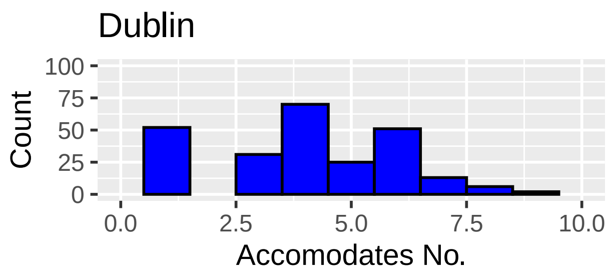
Here we can see that the majority of listings consist of 1-2 bedrooms. The amount of listings past 3 bedrooms are much more scarce for each city.

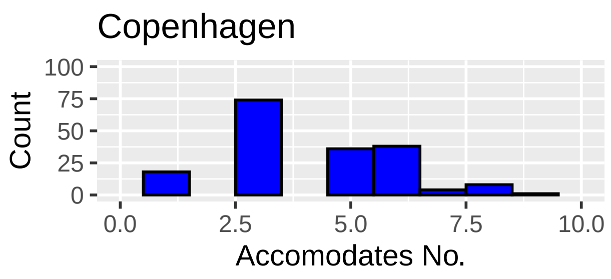
In regards to accommodation numbers,

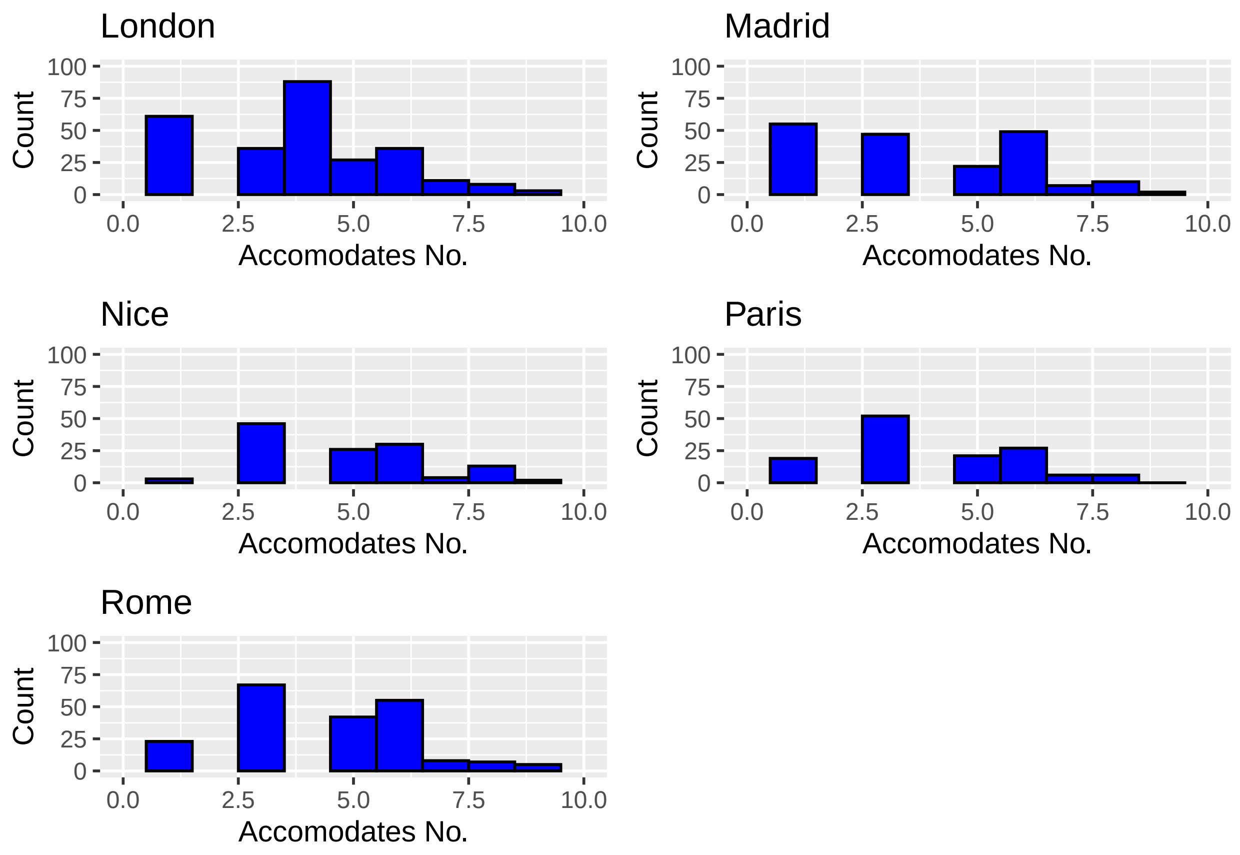

Amsterdam
Count
Accomodates No.


Barcelona
Count
Accomodates No.


Bordeaux
Count
Accomodates No.

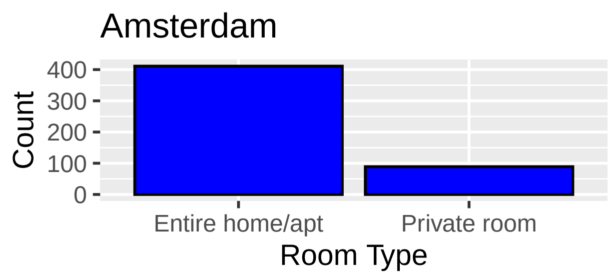

Dublin
Count
Accomodates No.

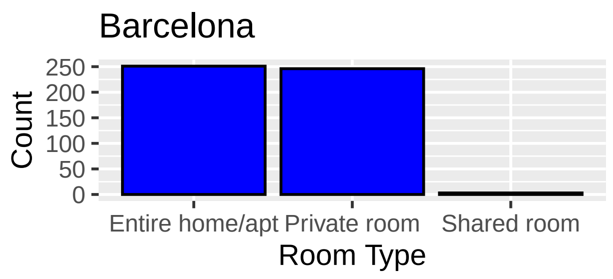

Copenhagen
Count
Accomodates No.

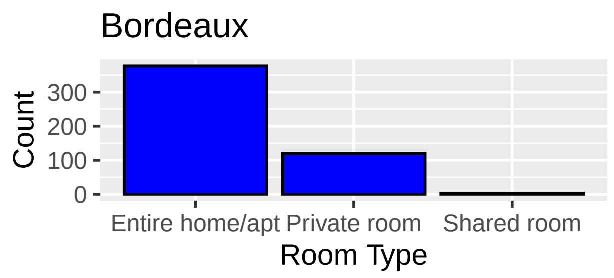


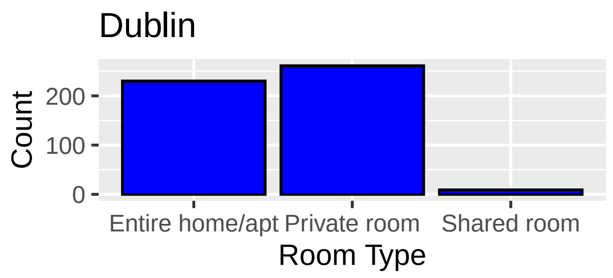
We can see that for all cities, the overall range of accomodation numbers for a listing spans from 1-10 persons. The most common accomodations numbers amongst all city listings are 1-2 OR 3-4 people.

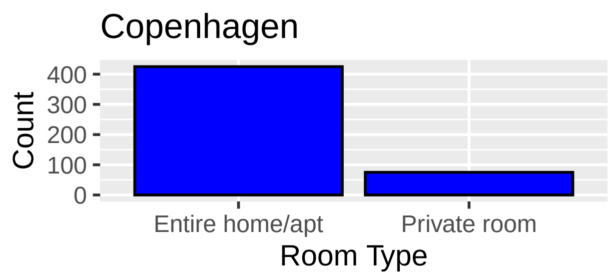
Now looking at room numbers,

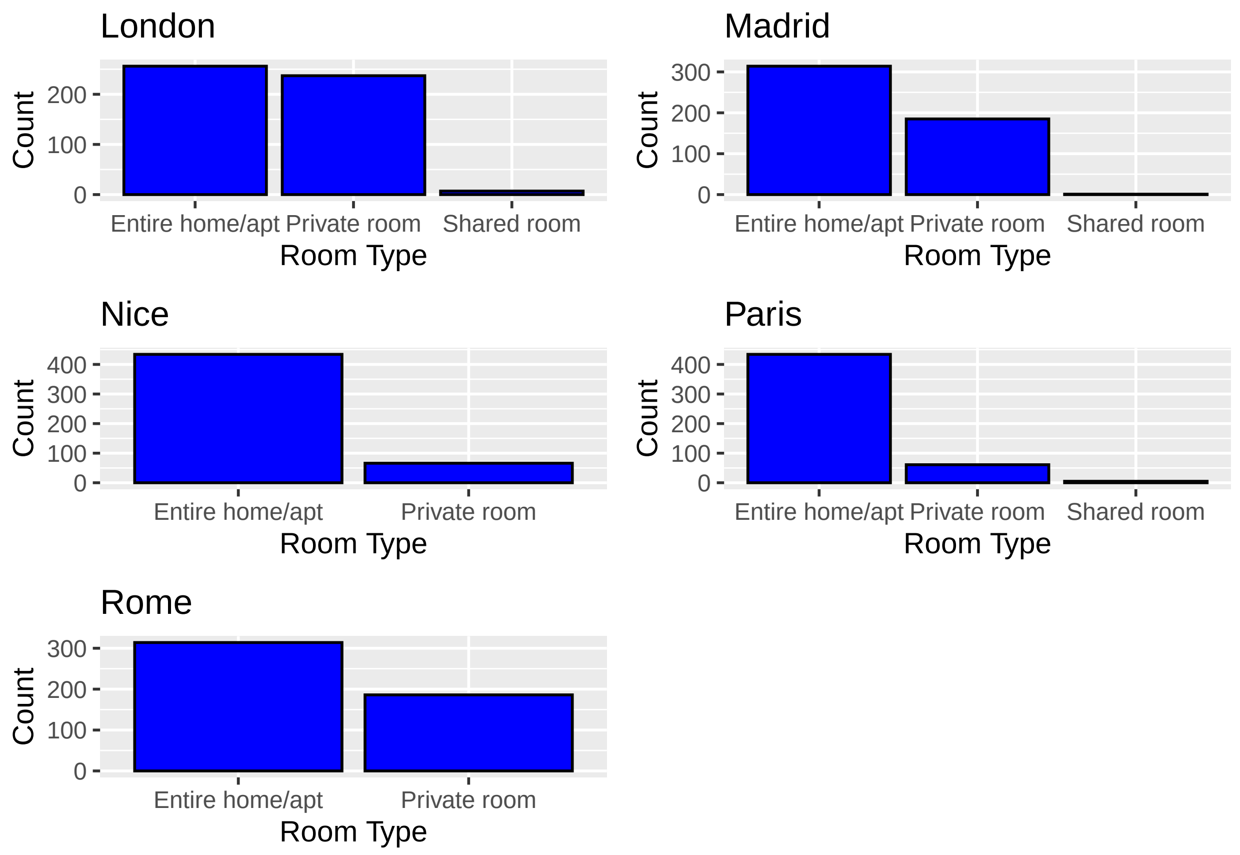

Amsterdam
Count
400
300
200
100
0
Entire home/apt
Private room
Room Type


Barcelona
Count
250
200
150
100
50
0
Entire home/apt
Private room
Shared room
Room Type


Bordeaux
Count
300
200
100
0
Entire home/apt
Private room
Shared room
Room Type


Dublin
Count
200
100
0
Entire home/apt
Private room
Shared room
Room Type


Copenhagen
Count
400
300
200
100
0
Entire home/apt
Private room
Room Type

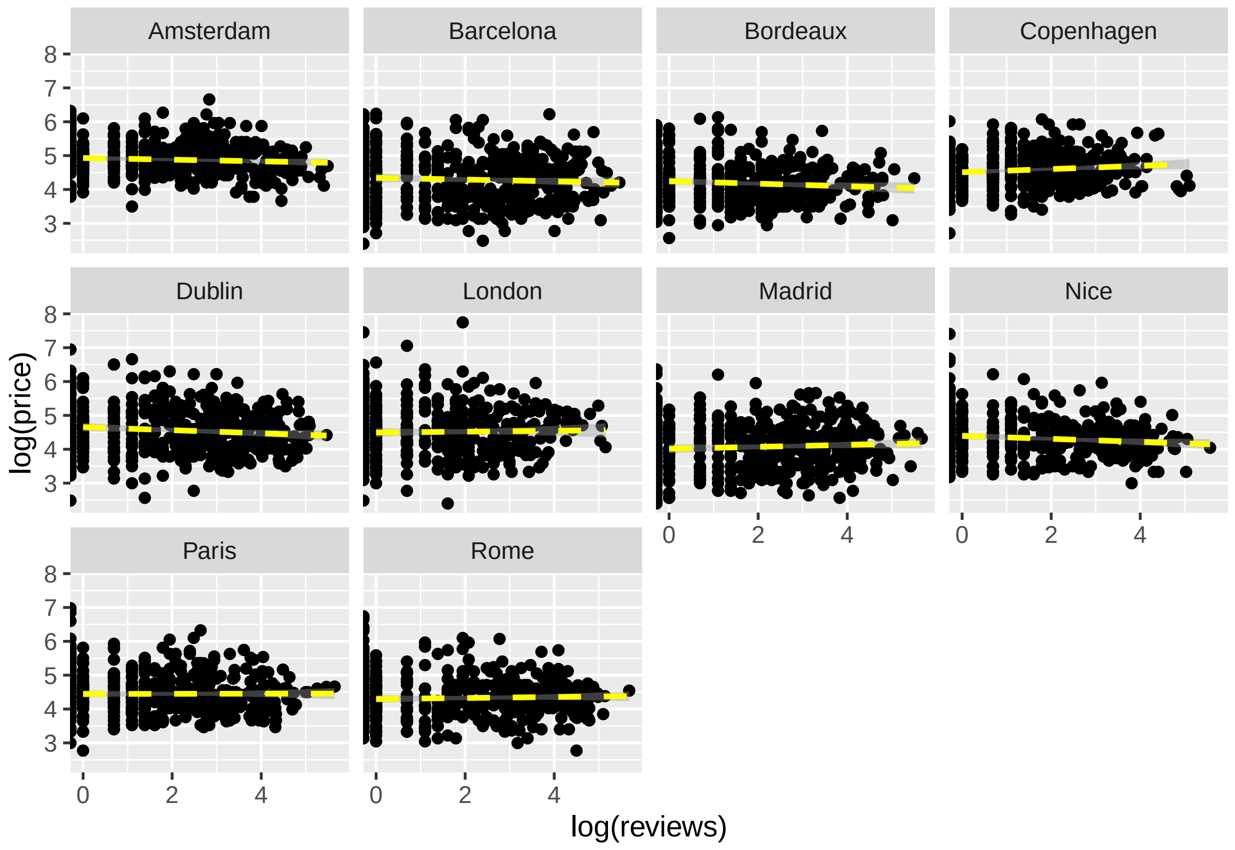


"Entire home/apartment" and "Private Rooms" are the most common listing room types overall. Paris, Amsterdam, and Copenhagen have the largest skew towards mainly the "Entire home/apartment" type.

EDA Part 2

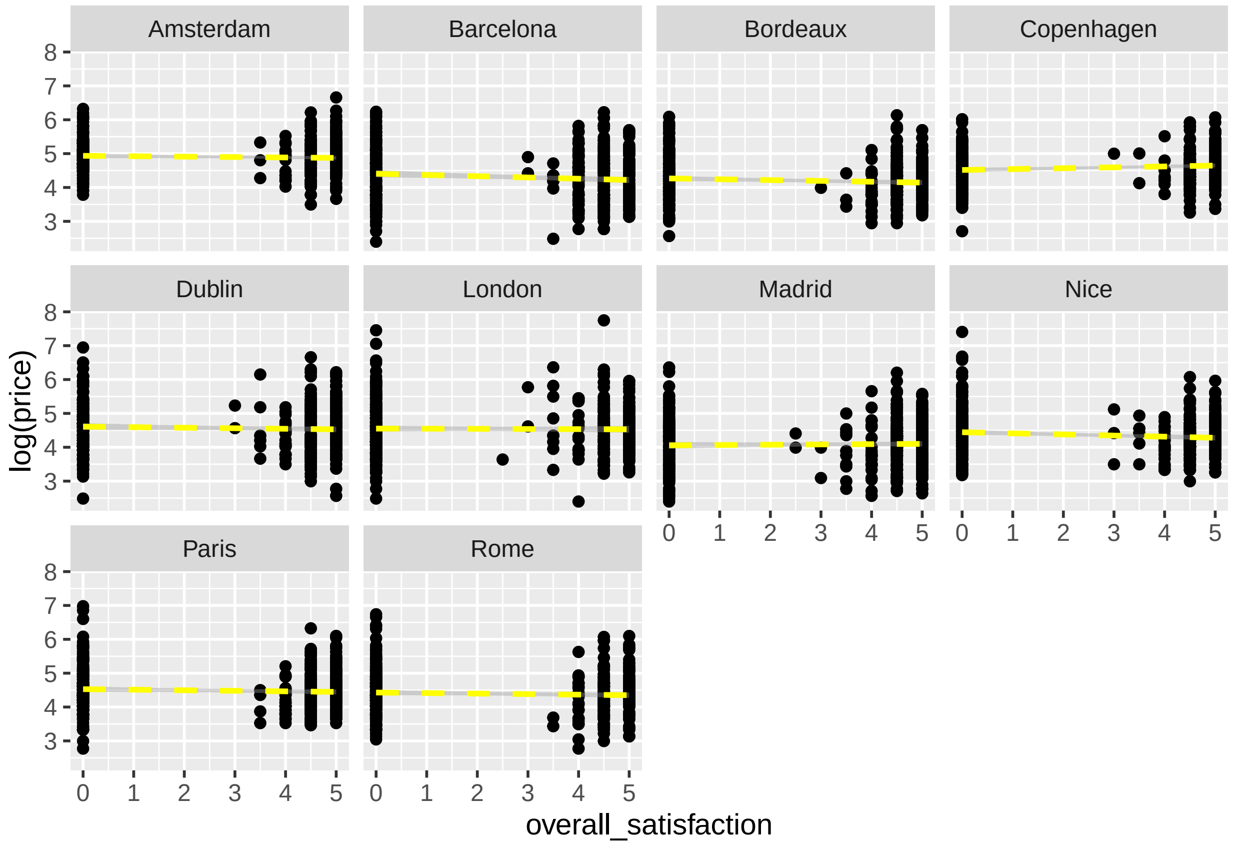
From here, we will begin to look to plot Price against a number of the group level predictor variables amongst each city, to see if we can determine the need for either random intercepts, random slopes, or both in our final multilevel model analysis.

Looking at review counts,



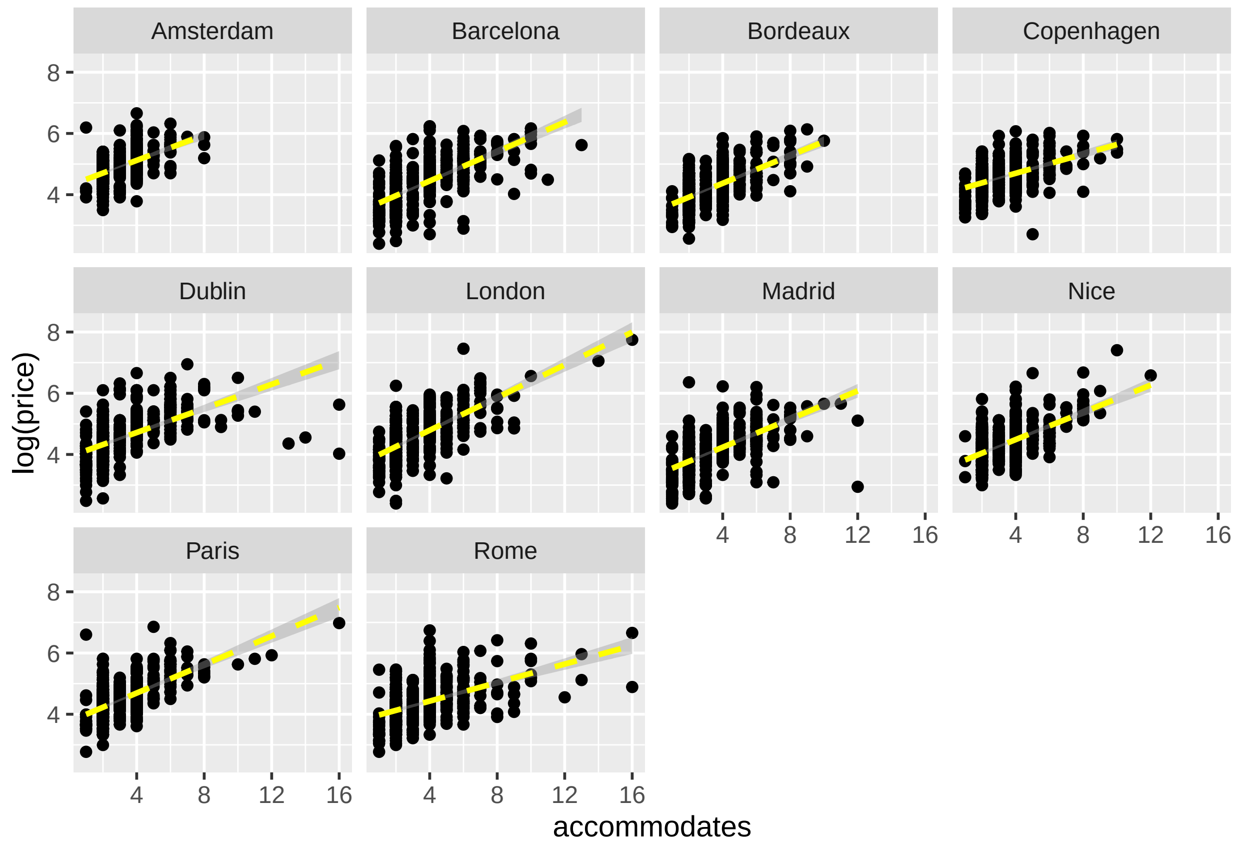
It looks as if in this case that review counts may not necessarily have much of an effect at all on the price outcome, however, due to the possibilities of confouding effects we will still include this as a predictor in our model.

Now looking at overall satisfaction,



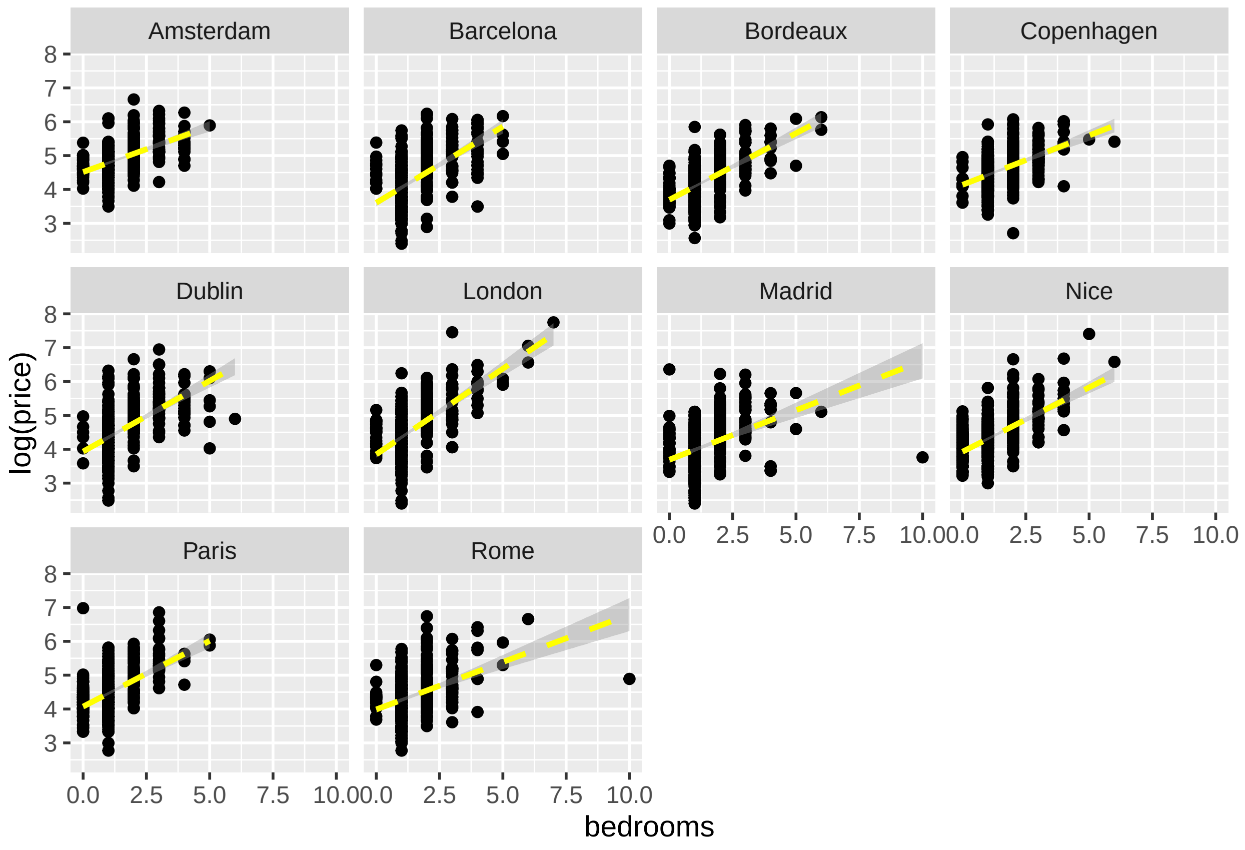
In addition, suprisingly enough overall satisfaction ratings also does not seem to have a strong effect on price outcomes, but similar to review counts, we will still control for this variable in our model.

Now looking at accomodations numbers,



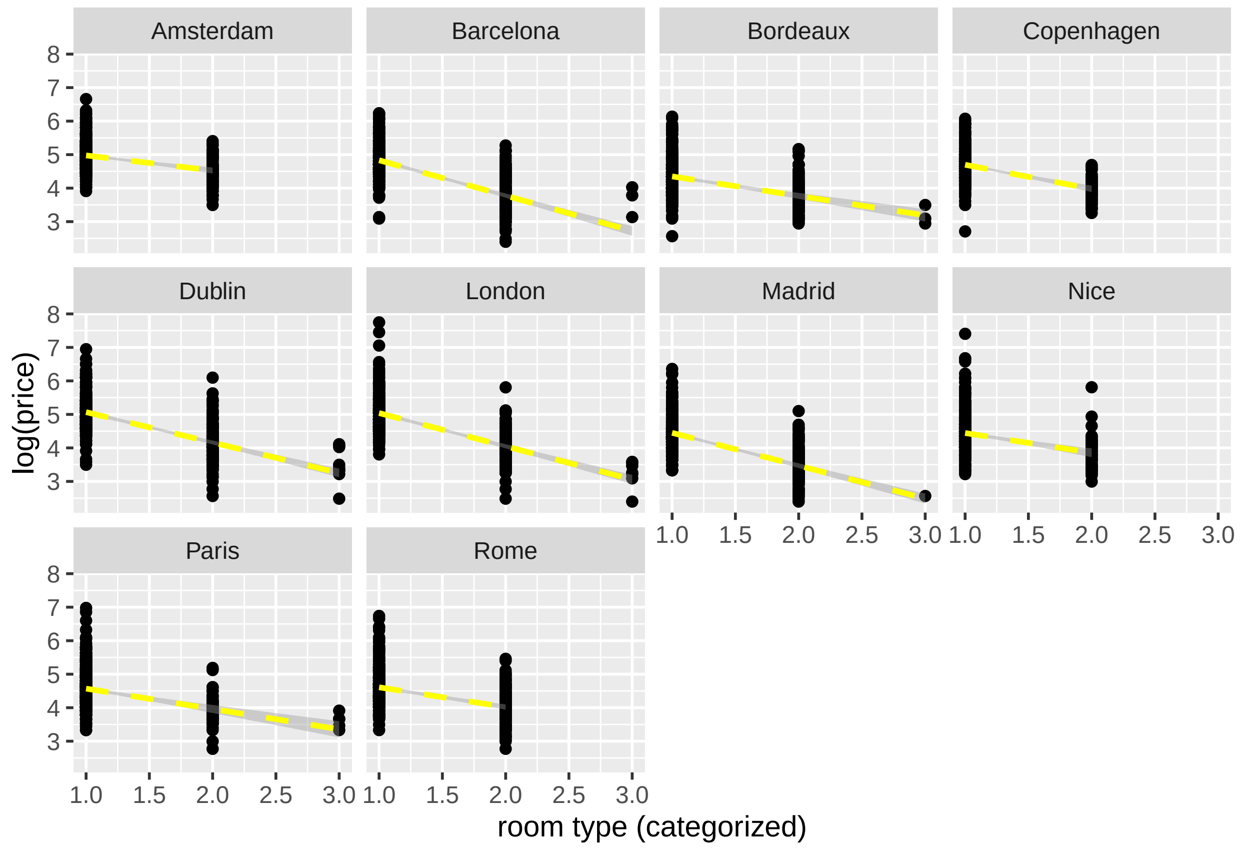
There is a clear positive effect between accomodation numbers, and listing prices. The slopes seem to vary amongst cities and therfore we may look to include random slope within our model.

Now looking at bedrooms,



Here there also looks to be a change in slope amongst cities, and therefore random slope for bedrooms will also be accounted for in the model.

Finally, room type.



Here we can see that prices for listings in all cities certainly fluctuate based on their room types, with “Shared” as the cheapest and “Entire home/apt” the most expensive. Room type will also be included as a predictor in our model, however to start, we will keep it as a fixed effect.

Multilevel Model Analysis

From our EDA above, evidence suggests that we should look to observe a random intercept, random slope model to be used in our modelling of AirBNB listing prices. Once we have fitted the model, we’ll check how well it models the data, and once we have determined an appropriate model, we will look to finally make predictions.

The random intercept in the model will be accounting for the difference between city groupings of AirBNB data, and the random slopes accounted for will be accommodation numbers, and number of bedrooms.

```
fit <- lmer(log(price) ~ reviews + overall_satisfaction + bedrooms
            + accommodates + room_type + (1 + bedrooms + accommodates| City),
            data = BNBdata)


summary(fit)$coefficients
```

```
##                           Estimate    Std. Error    t value
## (Intercept)           4.1556229135 0.0864706217  48.058206
## reviews              -0.0004398213 0.0002394055  -1.837139
## overall_satisfaction -0.0157128503 0.0030168321  -5.208394
## bedrooms               0.1507005296 0.0293023756   5.142946
## accommodates           0.0950333141 0.0112079654   8.479087
## room_typePrivate room -0.5368318212 0.0166832418 -32.177908
## room_typeShared room  -1.1052424743 0.0838147956 -13.186723
```

Interpretation of the multilevel model (need to update)

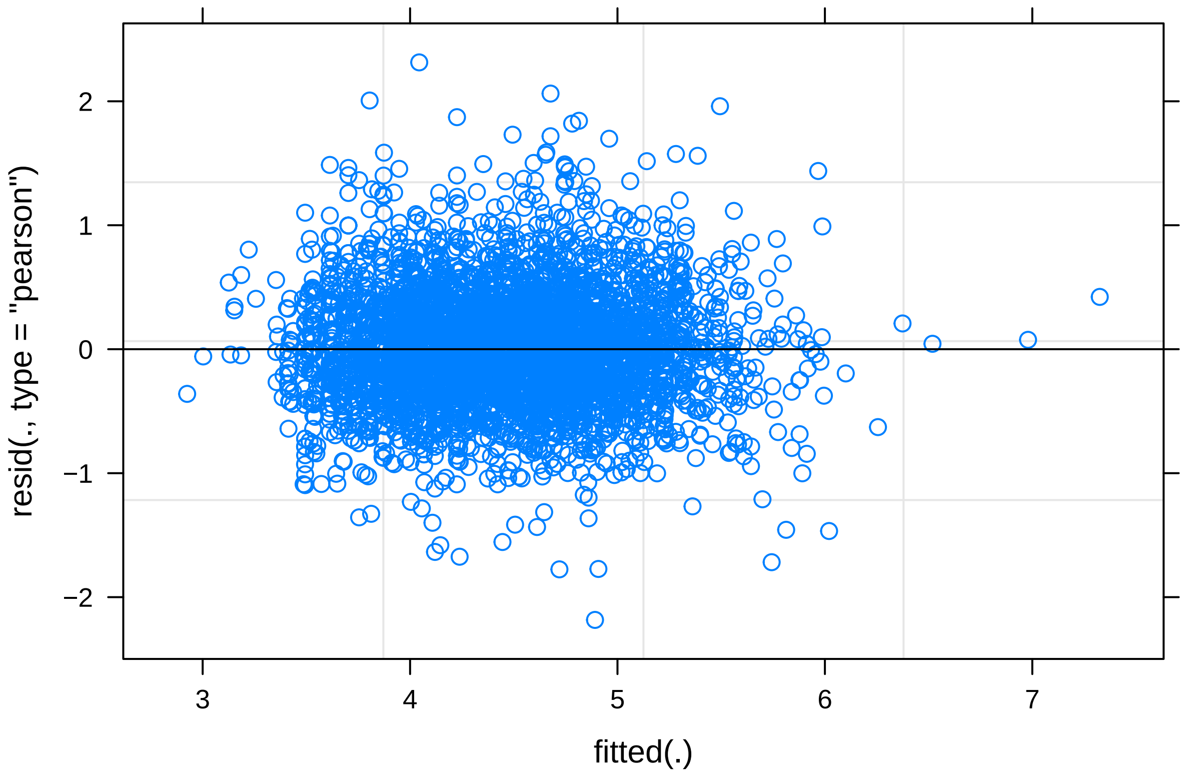
From the model coefficient estimates, we can see that the number of bedrooms and number of people being accommodated in a listing has the largest positive effect of price of the listing. Once we've taken the exponentials of the coefficient values (which we need to do as we centred to outcome price on the log scale), we can see that for every one extra bedroom in the listing, price is expected to increase around 16%. In addition, to every 1 extra person that a listing will accomodate, the price is expected to increase around 10%. For rooms types as well, booking a private room should on average cost you around 42% less than an Entire home/aprt. In addition, choosing to booked a shared room will cost you on average around 67% less than an Entire home/aprt.

The number of reviews actually has a negative effect here on price, which intuitively, would be against what we would expect, however from the EDA above, we could see that review counts and overall satisfaction does not actually hold much of an effect at all on price. We rather opted to account for both in the model for the sake of being dilligent.

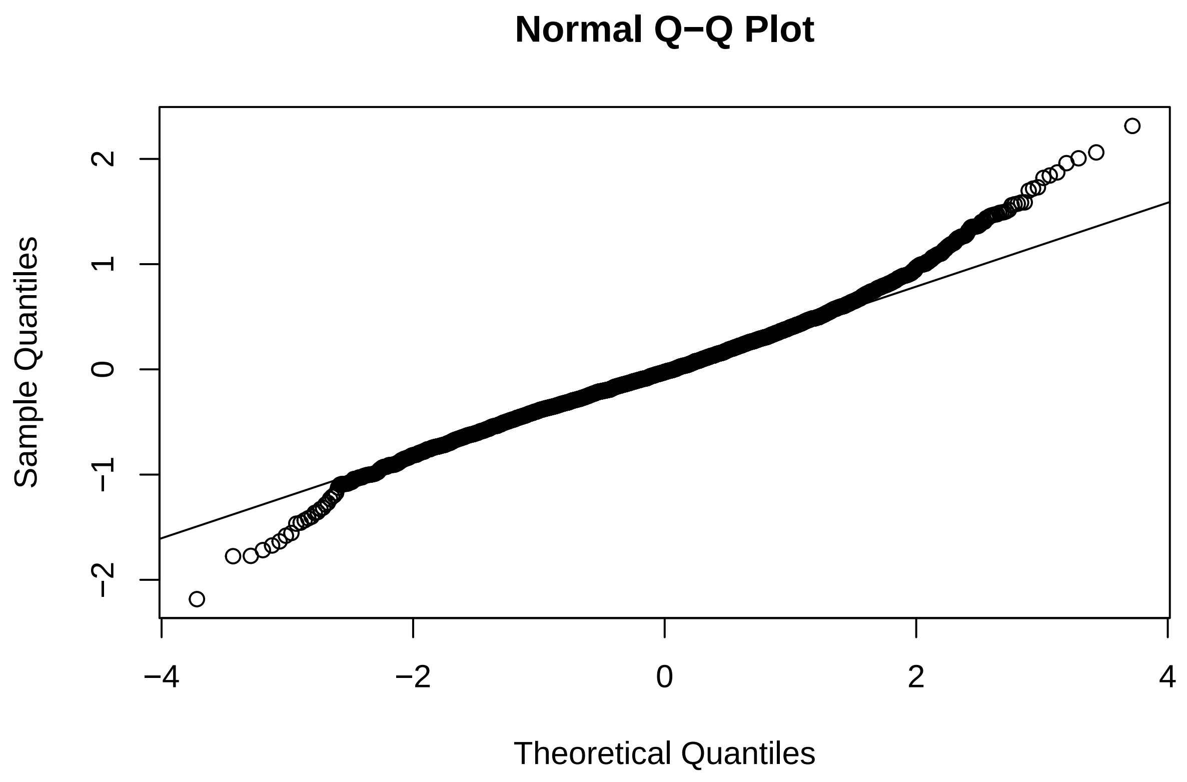
Further summary output of the model can be found in the Appendix.

What now needs to be done, is to check the fit of the model.

```r
plot(fit)
```



```r
qqnorm(resid(fit))
qqline(resid(fit))
```



We can see from this model check, that the majority of the residuals follow normality, however there are a few outliers also present (as seen from the tails of the qqplot) and it's important to note that this could have an affect on making predictions based on this model. It's possible that this AirBnB data set may have a higher number of extreme values than would be expected, if the data was truly from a normal distribution.

AirBnB Price Predictions

For the purpose of this analysis, we are now going to look to make AirBnB listing price predictions, for hypothetical new listings within the existing city groups from our data.

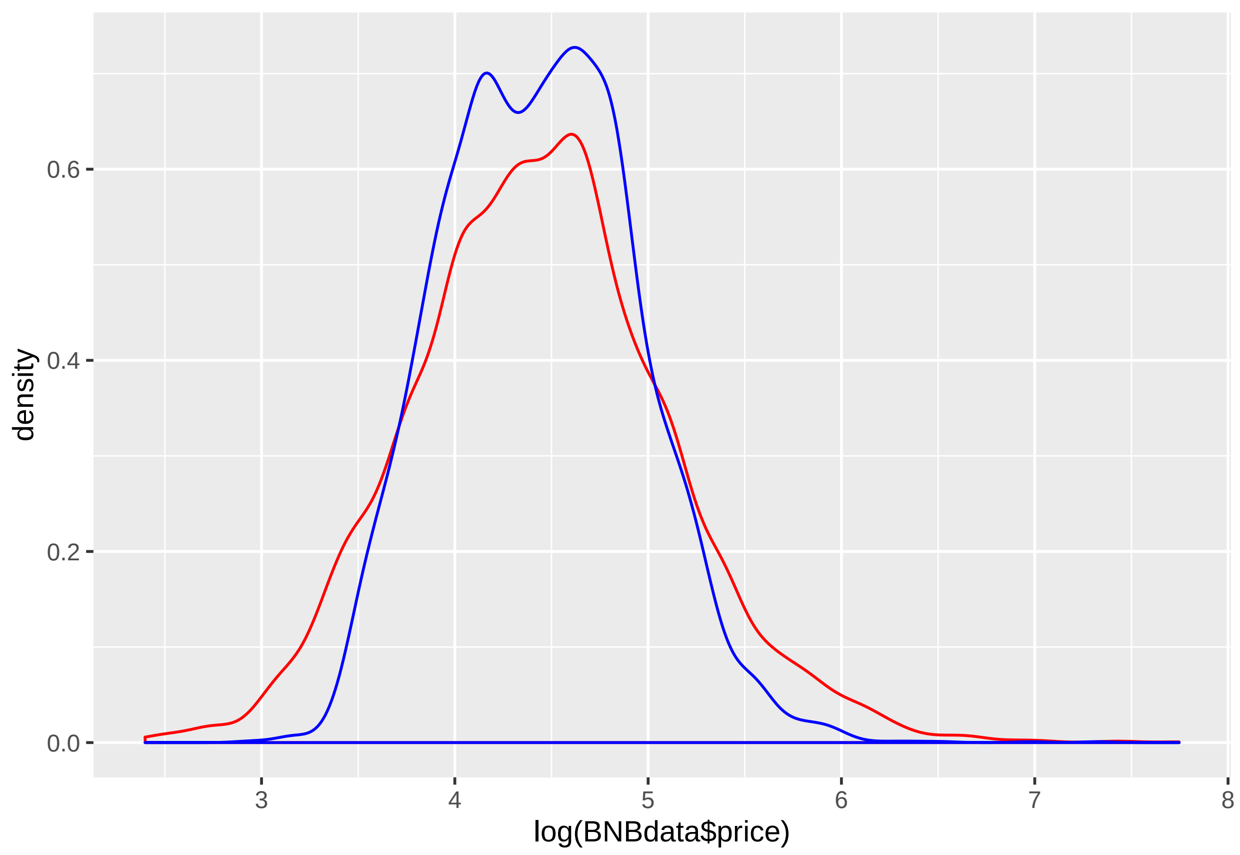
First to make predictions for each listing in the dataset at a hypothetical next time point.

```
## as the price outcome is in a log base for our model, we have to exponentiate the predictions.

predict_new <- exp(predict(fit,newdata=BNBdata))

predict_values <- cbind(BNBdata, predict_new)

ggplot(BNBdata) + geom_density(aes(x = log(BNBdata$price)), color = 'red') + geom_density(aes(x = log(pr
```

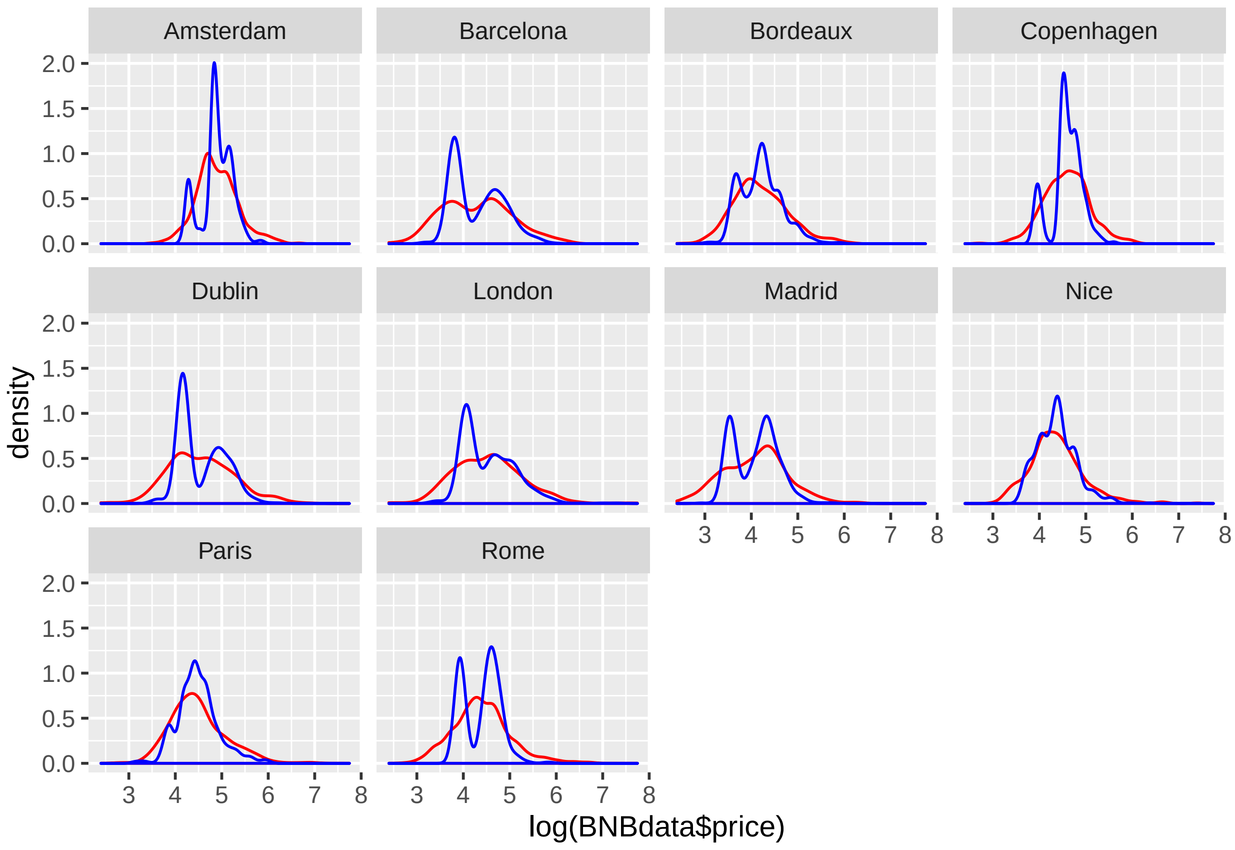


It is important to note, that given the slight bimodality shown in our prediction distribution from this model, I also tested the centering and scaling of predictors towards z scores in the model, in addition to removing the random slopes. This however only increased the bimodality of the predictions, and hence with no improvements I elected to stick with this fitting of the model as it has been the most efficient fit. In addition, if you look back to the density plots of price for each city, there are a few that tend to be bimodal and in this case, that is most likely causing for the pull towards slight bimodality in the predictions.

From here we can see that the prediction values (blue line) for prices of listing amongst all cities pooled are similar, but there is certainly variance present, as can be understood from the residual checks above.

To look more deeply into the prediction density amongst cities,

```r
ggplot(BNBdata) + geom_density(aes(x = log(BNBdata$price)),
  color = 'red') + geom_density(aes(x = log(predict_values$predict_new)),
                    color = 'blue') + facet_wrap(~City)
```



Again we can see, the predictoin model tends to create some outlier predictions, hence the bimodality we see. In regards to city, this model seems to best predict for listings in Nice, Paris, and Bordeaux.

Once we have a model fit as such, we now have the ability to make predictions based on specific scenarios of listings with a particular set of attributes. The implications of this, and what it could mean for a company like AirBNB to have strong prediction models run on their data, will be discussed below.

Results and Discussion

Given this multilevel model fit, we can see how predictions can be made on listing prices based on their City, accommodations, bedrooms, reviews, and room types attributes. Models like such are useful for companies like AirBNB to provide better services to their customers, by giving estimations on what listings will cost based on what the customer is looking for. This would give the customer the opportunity to see how the prices should generally vary based on listing attributes and see clear trends before researching into the actual listings. As searching through specific listings to generate an idea of what attributes will affect price more/less, can be a time consuming process.

In addition, models like this are also particularly useful for the host side as well. A host could look to put up a listing, submit their listing's attributes and AirBNB could then make suggestions on what reasonable prices the host should put their listing up for, based on the prices that have been successful for similar listings. This would ideally help eliminate any lag between host's posting listings, and getting their price to a value that will be agreed upon from the customer, without the host having to go out and research through AirBNB listings themselves.

There are also a few ways in which we could look to further test this above model and make improvements. Earlier we saw that the residual checks to the fitted multilevel model look appropriate, with City as our random intercept, accommodates and bedrooms as random slopes, along with room type, overall satisfaction and review count set as fixed effects. However in the predictions there are a few overestimations, as well as a trend of bimodality amongst respective city predictions. This could possibly be due to the bimodality that also could be seen during the EDA in the price distributions amongst cities. It may be useful to further extend this model in the future to other available city data from AirBNB listings, to see if this trend continues in additional cities, or if there is varying results. This would allow us to further evaluate the data and make determinations on even stronger ways to then fit the predictive models.

Appendix

Reading in Data, Cleaning, and Combining

For the purpose of this analysis, our city data sets were sampled from the raw AirBNB public city data sets. Each raw data set recorded their respective city's listings within the time period of February - April of 2017.

The sampling of the raw data sets was done on an external R script, that is also in the same Github repo if there is interest to look into how the data was captured.

Firstly, we read in our data for each of the 10 cities.

```
read <- function(data){

  return(read.csv(data, header = T))
}

Amsterdam <- read("Amsterdam.csv")
Barcelona <- read("Barcelona.csv")
Bordeaux <- read("Bordeaux.csv")
Dublin <- read("Dublin.csv")
Copenhagen <- read("Copenhagen.csv")
London <- read("London.csv")
Madrid <- read("Madrid.csv")
Nice <- read("Nice.csv")
Paris <- read("Paris.csv")
Rome <- read("Rome.csv")
```

Now, our goal will be to combine these data sets together for analysis and so we need to ensure that each data set has the same dimensions, and variable distinction.

```
df <- data.frame(dim(Amsterdam), dim(Barcelona), dim(Bordeaux), dim(Dublin), dim(Copenhagen), dim(Londo

df
```

```
##   dim.Amsterdam. dim.Barcelona. dim.Bordeaux. dim.Dublin. dim.Copenhagen.
## 1            500            500           500         500             500
## 2             15             15            15          15              15
##   dim.London. dim.Madrid. dim.Nice. dim.Paris. dim.Rome.
## 1         500         500       500        500       500
## 2          15          15        15         15        15
```

```
## Example check for variable equality
identical(colnames(Copenhagen), colnames(Amsterdam))
```

```
## [1] TRUE
```

Through multiple variable equality checks, it was found that each data set has the same dimensions and variable designations, therefore we should be able to combine these city data sets fairly easily.

Now that we have each city data set with the same dimensions, and corresponding column values, the last thing to do prior to combining cities and to add a city column to each data set, so that the observations can be grouped once combined.

```r
addCity <- function(df, city){
  return(mutate(df, City = city))
}

Amsterdam <- addCity(Amsterdam, "Amsterdam")
Barcelona <- addCity(Barcelona, "Barcelona")
Bordeaux <- addCity(Bordeaux, "Bordeaux")
Dublin <- addCity(Dublin, "Dublin")
Copenhagen <- addCity(Copenhagen, "Copenhagen")
London <- addCity(London, "London")
Madrid <- addCity(Madrid, "Madrid")
Nice <- addCity(Nice, "Nice")
Paris <- addCity(Paris, "Paris")
Rome <- addCity(Rome, "Rome")
```

We can now combine.

```r
BNBdata <- rbind(Amsterdam, Barcelona, Bordeaux, Dublin, Copenhagen,
                 London, Madrid, Nice, Paris, Rome)
```

Now, borough and minstay column values are NA, so we are going to immediately drop these.

```r
BNBdata <- dplyr::select(BNBdata, -c(minstay, borough))
```

We also want to check to see if there are any other NA values that we've missed. Of course, given the random sampling at the beginning of this analysis, there is the possibility of these results varying each time the analysis is reproduced, however it should not affect results as long as the NA values are low (which has been confirmed through inspecting the raw data).

```r
sapply(BNBdata, function(x) sum(is.na(x)))
```

```
##                    X              room_id              host_id
##                    0                    0                    0
##            room_type         neighborhood              reviews
##                    0                    0                    0
## overall_satisfaction         accommodates             bedrooms
##                    0                    0                    0
##                price             latitude            longitude
##                    0                    0                    0
##        last_modified                 City
##                    0                    0
```

Now double checking for NA values, and dropping the last variable neighborhood that will not be needed for analysis.

```r
BNBdata <- dplyr::select(BNBdata, -neighborhood)

BNBdata <- na.omit(BNBdata)
dim(BNBdata)
```

```
## [1] 5000   13
```

```r
BNBdata <- dplyr::select(BNBdata, -c(longitude, latitude, last_modified))

head(BNBdata)
```

```
##   X  room_id   host_id       room_type reviews overall_satisfaction
## 1 1  7005410  34197767 Entire home/apt       3                  4.0
## 2 2  7854067   6870134    Private room     109                  5.0
## 3 3 17279326  68362369 Entire home/apt       0                  0.0
## 4 4 16235663 106117460 Entire home/apt       5                  4.5
## 5 5  6905082  25676251 Entire home/apt       0                  0.0
## 6 6  3017539   4389536 Entire home/apt       1                  0.0
##   accommodates bedrooms price      City
## 1            4        2   250 Amsterdam
## 2            2        0   131 Amsterdam
## 3            4        2   156 Amsterdam
## 4            3        0   128 Amsterdam
## 5            2        1   203 Amsterdam
## 6            4        2   200 Amsterdam
```

EDA Code (corresponding to visualizations in report)

EDA Part 1

Density Plots for Price.

```
denseplot <- function(data, city1, city2, city3, city4, city5)
{
  half_1 <- filter(data, data$City == city1 | data$City == city2 | data$City == city3
                   | data$City == city4 | data$City == city5)

  ggplot(half_1) + geom_density(aes(x = log(price), color = City),    size = 1) + xlim(0,8)
}

a <- denseplot(BNBdata, "Amsterdam", "Barcelona", "Bordeaux", "Copenhagen", "Dublin")
b <- denseplot(BNBdata, "London", "Madrid", "Nice", "Paris", "Rome")

## splitting cities into two density graphs, so that the city plots are easier to distinguish
grid.arrange(a, b, ncol = 1)
```

City listing bedroom number histograms.

```
bedHist <- function(data, city) {

  dat <- filter(data, City == city)
  ggplot(dat) + geom_histogram(aes(x=dat$bedrooms), binwidth=1, col="black",
                     fill="blue") +
    ylab("Count") + xlab("Bedrooms") + ggtitle(city) + xlim(0,10) + ylim(0,100)
}


a <- bedHist(BNBdata, "Amsterdam")
b <- bedHist(BNBdata, "Barcelona")
c <- bedHist(BNBdata, "Bordeaux")
d <- bedHist(BNBdata, "Dublin")
e <- bedHist(BNBdata, "Copenhagen")
f <- bedHist(BNBdata, "London")
g <- bedHist(BNBdata, "Madrid")
h <- bedHist(BNBdata, "Nice")
i <- bedHist(BNBdata, "Paris")
```

```r
j <- bedHist(BNBdata, "Rome")

grid.arrange(a,b,c,d,e, ncol = 2)
grid.arrange(f,g,h,i,j, ncol = 2)
```

City listing accommodation histograms.

```r
accomHist <- function(data, city) {

  dat <- filter(data, City == city)
  ggplot(dat) + geom_bar(aes(x=dat$accommodates), binwidth = 1, col="black",
                         fill="blue") +
    ylab("Count") + xlab("Accomodates No.") + ggtitle(city) + ylim(0, 100) + xlim(0,10)
}

a <- accomHist(BNBdata, "Amsterdam")
b <- accomHist(BNBdata, "Barcelona")
c <- accomHist(BNBdata, "Bordeaux")
d <- accomHist(BNBdata, "Dublin")
e <- accomHist(BNBdata, "Copenhagen")
f <- accomHist(BNBdata, "London")
g <- accomHist(BNBdata, "Madrid")
h <- accomHist(BNBdata, "Nice")
i <- accomHist(BNBdata, "Paris")
j <- accomHist(BNBdata, "Rome")

grid.arrange(a,b,c,d,e, ncol = 2)
grid.arrange(f,g,h,i,j, ncol = 2)
```

City listing room number histograms.

```r
roomHist <- function(data, city) {

  dat <- filter(data, City == city)
  ggplot(dat) + geom_bar(aes(x=dat$room_type), col="black",
                         fill="blue") +
    ylab("Count") + xlab("Room Type") + ggtitle(city)
}

a <- roomHist(BNBdata, "Amsterdam")
b <- roomHist(BNBdata, "Barcelona")
c <- roomHist(BNBdata, "Bordeaux")
d <- roomHist(BNBdata, "Dublin")
e <- roomHist(BNBdata, "Copenhagen")
f <- roomHist(BNBdata, "London")
g <- roomHist(BNBdata, "Madrid")
h <- roomHist(BNBdata, "Nice")
i <- roomHist(BNBdata, "Paris")
j <- roomHist(BNBdata, "Rome")

grid.arrange(a,b,c,d,e, ncol = 2)
grid.arrange(f,g,h,i,j, ncol = 2)
```

EDA Part 2

Review counts vs. price plotting.

```
## reviews

ggplot(BNBdata, aes(x = log(reviews), y = log(price))) + geom_point() + geom_smooth(method='lm', linety
```

Overall satisfaction vs. price plotting.

```
ggplot(BNBdata, aes(x = overall_satisfaction, y = log(price))) + geom_point() + geom_smooth(method='lm'
```

Accomodation numbers vs. price plotting.

```
ggplot(BNBdata, aes(x = accommodates, y = log(price))) + geom_point() + geom_smooth(method='lm', linety
```

Bedroom numbers vs. price plotting.

```
ggplot(BNBdata, aes(x = bedrooms, y = log(price))) + geom_point() + geom_smooth(method='lm', linetype =
```

Room types vs. price plotting.

```
BNBdata_room <- BNBdata %>% mutate(room_num = NA)
BNBdata_room$room_num[BNBdata_room$room_type == "Entire home/apt"] <- 1
BNBdata_room$room_num[BNBdata_room$room_type == "Private room"] <- 2
BNBdata_room$room_num[BNBdata_room$room_type == "Shared room"] <- 3

ggplot(BNBdata_room, aes(x = room_num, y = log(price))) + geom_point() + geom_smooth(method='lm', linet
```

Full Model Summary Output

```
fit <- lmer(log(price) ~ reviews + overall_satisfaction +
              bedrooms + accommodates + room_type +
              (1 + bedrooms + accommodates| City), data = BNBdata)


summary(fit)
```

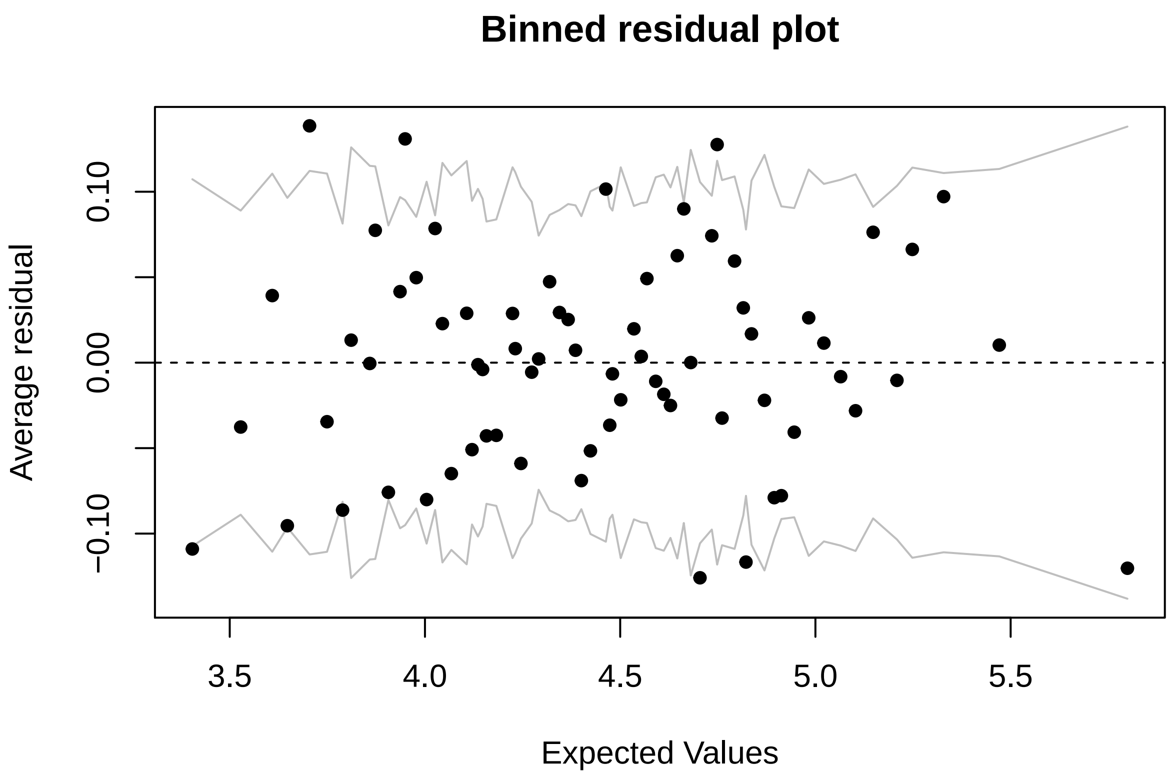
```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## log(price) ~ reviews + overall_satisfaction + bedrooms + accommodates +
##     room_type + (1 + bedrooms + accommodates | City)
##    Data: BNBdata
##
## REML criterion at convergence: 6029.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.0042 -0.6400 -0.0575  0.5923  5.3022
##
## Random effects:
##  Groups   Name         Variance  Std.Dev. Corr
##  City     (Intercept)  0.0711465 0.26673
##           bedrooms     0.0074657 0.08640  -0.54
##           accommodates 0.0009158 0.03026   0.02 -0.36
##  Residual              0.1904038 0.43635
## Number of obs: 5000, groups:  City, 10
```

```
##
## Fixed effects:
##                         Estimate Std. Error t value
## (Intercept)            4.1556229  0.0864706  48.058
## reviews               -0.0004398  0.0002394  -1.837
## overall_satisfaction  -0.0157129  0.0030168  -5.208
## bedrooms               0.1507005  0.0293024   5.143
## accommodates           0.0950333  0.0112080   8.479
## room_typePrivate room -0.5368318  0.0166832 -32.178
## room_typeShared room  -1.1052425  0.0838148 -13.187
##
## Correlation of Fixed Effects:
##             (Intr) reviws ovrll_ bedrms accmmd rm_tPr
## reviews     -0.002
## ovrll_stsfc -0.075 -0.434
## bedrooms    -0.495  0.029  0.010
## accommodats -0.043 -0.031 -0.016 -0.418
## rm_typPrvtr -0.131 -0.028  0.023 -0.034  0.200
## rm_typShrdr -0.028 -0.001  0.014 -0.008  0.042  0.117
## convergence code: 0
## Model failed to converge with max|grad| = 0.00610558 (tol = 0.002, component 1)
```

Additional Model Checks

Binned Residual Plot.

```
binnedplot(predict(fit), resid(fit, type = "response"))
```


Binned residual plot
Average residual
0.10
0.00
−0.10
3.5
4.0
4.5
5.0
5.5
Expected Values