ISCTE-IUL
Licenciatura de Ciência de Dados

TODO Título

TODO Subtítulo

André Plancha, 105289
<Andre_Plancha@iscte-iul.pt>
Tomás Ribeiro, 105220
<tfroo1@iscte-iul.pt>
Afonso Silva, 105208
<agsos@iscte-iul.pt>
Rui Chaves, 104914
<rfpcs1@iscte-iul.pt>

21/12/2022
Versão 0.0.2

```
## package 'Metrics' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\tomas\AppData\Local\Temp\RtmpqmkfSZ\downloaded_packages
##
##   There is a binary version available but the source version is later:
##        binary source needs_compilation
## devEMF  4.1-1  4.1-2              TRUE
##
##   Binaries will be installed
## package 'gsubfn' successfully unpacked and MD5 sums checked
## package 'proto' successfully unpacked and MD5 sums checked
## package 'RSQLite' successfully unpacked and MD5 sums checked
## package 'chron' successfully unpacked and MD5 sums checked
## package 'sqldf' successfully unpacked and MD5 sums checked
## package 'data.table' successfully unpacked and MD5 sums checked
## package 'devEMF' successfully unpacked and MD5 sums checked
```

```
## package 'Fgmutils' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\tomas\AppData\Local\Temp\RtmpqmkfSZ\downloaded_packages
```

```
df <- read.csv(here("data", "listings.csv"))
shape <- st_read(here("data", "SF Planning Neighborhood Groups Map"))
tmap_mode("plot")
shape_plot <- shape %>% ggplot()+ geom_sf() + theme(legend.position = "bottom")
```

A base de dados que nos foi disponibilizada vem do projeto, fundado por Murray Cox com a missão de "[...] fornecer dados e defesa sobre o impacto do Airbnb em comunidades residenciais" [2].

A base de dados contém 6629 entradas, e cada uma delas representa um registo de um anúncio para o aluguer de um alojamento disponível no Airbnb, em São Francisco, Califórnia. Cada alojamento contém informação sobre o seu preço, localização, hospedeiro, o tipo de alojamento, as *reviews* do alojamento, e licensa do alojamento.

```
data.frame(
  row.names = colnames(df),
  "type" = sapply(df, class)
) %>% showT()
```

	type
id	numeric
name	character
host_id	integer
host_name	character
neighbourhood_group	logical
neighbourhood	character
latitude	numeric
longitude	numeric
room_type	character
price	integer
minimum_nights	integer
number_of_reviews	integer
last_review	character
reviews_per_month	numeric
calculated_host_listings_count	integer
availability_365	integer
number_of_reviews_ltm	integer
license	character

De forma a perceber melhor a base dados, o *Airbnb* disponibiliza de um "dicionário de dados" [1] que explica o significado de cada uma das variáveis:

- **id**: Número que representa um identificador único do anúncio;
- **name**: Título do anúncio;
- **host_id**: Identificador único da conta do hospedeiro;
- **host_name**: Nome da conta do hospedeiro (Normalmente este campo inclui apenas o primeiro nome ou nome da instituição hospedeira);

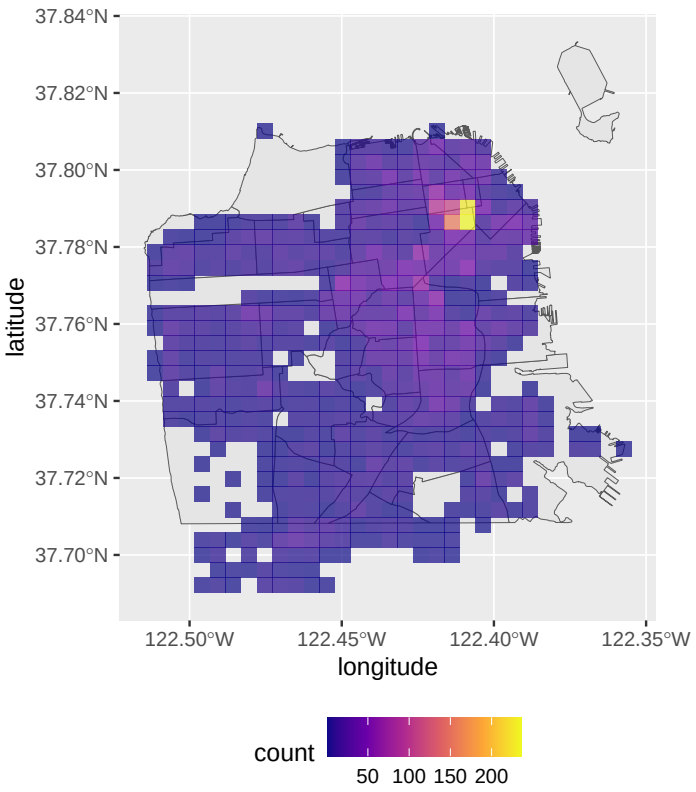
- **neighbourhood_group**: Este campo encontra-se vazio e não inclui descrição no dicionário;
- **neighbourhood**: Embora este campo não inclua descrição no dicionário, nesta base de dados este campo representa os bairros de São Francisco como definido pelo Departamento de Planeamento da cidade (os bairros de São Francisco não contém fronteiras oficiais e dependem da fonte (`tldrify.com/19p8`), logo a definição das fronteiras definidas pelo Airbnb tiveram de ser determinadas; mais à frente será demonstrado as fronteiras);
- **latitude/longitude**: Coordenadas geográficas do alojamento;
- **room_type**: Tipo de alojamento, entre "Quarto privado", "Quarto partilhado", "Quarto de hotel", e "Casa/Apartamento inteiro";
- **price**: Preço do alojamento por noite em USD;
- **minimum_nights**: Número mínimo de noites que o hospedeiro exige para alugar o alojamento;
- **number_of_reviews**: Número total de *reviews* que o alojamento tem desde o seu registo no Airbnb;
- **last_review**: Data da última *review* que o alojamento recebeu;
- **reviews_per_month**: Número médio de *reviews* que o alojamento recebe por mês;
- **calculated_host_listings_count**: Número de alojamentos que o hospedeiro tem disponíveis em São Francisco;
- **availability_365**: Número de dias que o alojamento está disponível por ano.
- **number_of_reviews_ltm**: Número de *reviews* que o alojamento recebeu nos últimos 12 meses;
- **license**: A licença/autorização/número de registo do alojamento.

Para o nosso objetivo, algumas colunas não vão ser úteis, devido à sua naturaza. Estas são o id, name, as categorias que referem informações sobre o hóspede (estas colunas conseguem justificar valores atípicos, principalmente em termos de preço; e.g. Um preço extremamente alto pode acontecer devido a um hotel de luxo na cidade. Estes problemas vão ser discutidos mais à frente.), a disponibilidade do alojamento durante o ano, e a licença do alojamento. Como o nosso objetivo será prever o preço esperado baseado na localização do apartamento, não vamos também utilizar variáveis associadas aos hóspedes, como o número de *reviews* e o número de alojamentos que o hóspede tem disponíveis em São Francisco. Cada registo contém as coordenadas geográficas e se as representármos graficamente, podemos verificar que grande parte dos alojamentos se encontram concentrados a nordeste da cidade, principalmente em *Downtown/Civic Center*. Apesar disso, também existem muitos alojamentos no resto da cidade.

Inesperadamente, o mapa mostra alguns pontos de alojamento fora da cidade, mas julgamos que não vá interferir com as nossas análises, devido ao correto agrupamento (demonstrado mais à frente) e à proximidade da cidade. Embora a razão nos seja desconhecida, acreditamos que o próprio Airbnb agrupa desta forma esses locais devido à sua proximidade com a cidade.
A concentração torna-se mais óbvia quando visualizamos o mapa de calor.

```
rast <- (shape_plot +
  stat_bin2d(
    data=df,
    aes(x=longitude, y=latitude),
    alpha=0.7, bins = 30, linejoin="round"
  ) +
  scale_fill_viridis_c(option="C"))
```

```
rast
```



O mapa claramente demonstra a concentração de alojamentos na zona clara, mas também consegue-se observar uma grande quantidade, embora mais dispersos, na zona central. Este gráfico demonstra uma possibilidade de agrupar os alojamentos nestas zonas.

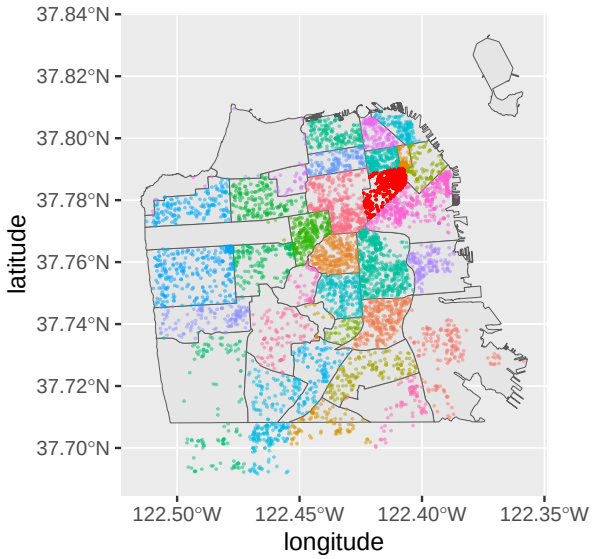
```
df %>% group_by(neighbourhood) %>% summarise(n=n(), freq = n/nrow(df)) %>%
  arrange(-n) %>% head(8) %>% showT()
```

neighbourhood	n	freq
Downtown/Civic Center	745	0.1123850
Mission	558	0.0841756
South of Market	450	0.0678835
Western Addition	418	0.0630563
Nob Hill	328	0.0494796
Outer Sunset	281	0.0423895
Bernal Heights	280	0.0422386
Haight Ashbury	276	0.0416352

A tabela mostra que grande parte dos alojamentos listados estão localizados no distrito de *Downtown/Civic Center* e *Mission*.

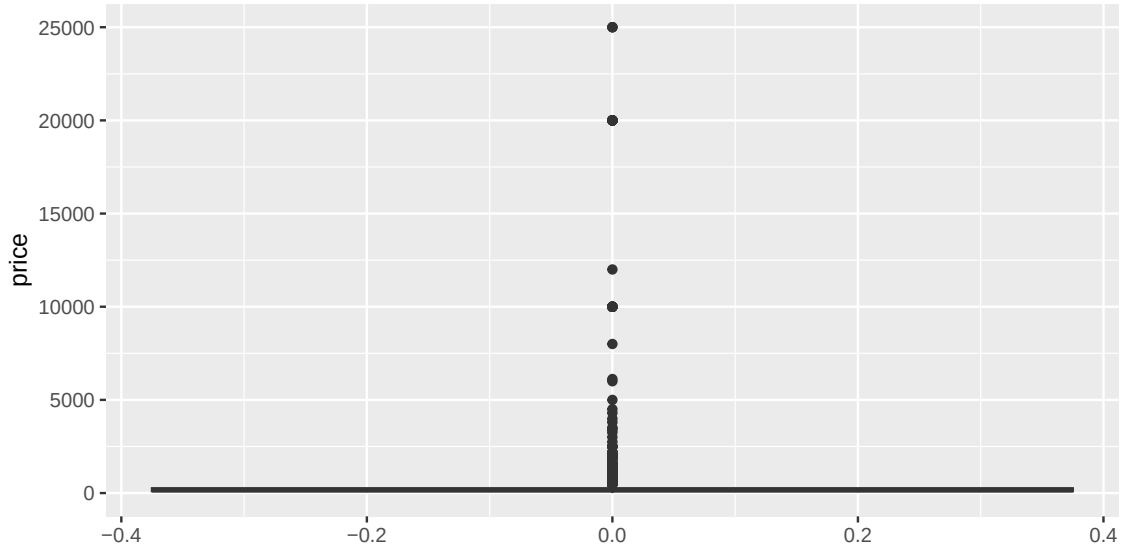
```
shape_plot +
  geom_point(data=df, aes(y=latitude, x=longitude, color=neighbourhood), alpha=0.5, size=
  geom_point(
    data=(df %>% filter(neighbourhood == "Downtown/Civic Center")),
```

```
  aes(y=latitude, x=longitude), color="red", alpha=1, size=0.1) +
  theme(legend.position = "none")
```



Esta figura demonstra que os bairros estão em conformidade com a definição do Departamento de Planejamento da cidade. Mostra também a posição do distrito *Downtown/Civic Center* a vermelho, através do mapa de calor.

```
ggplot(data=df, aes(price)) +
  geom_boxplot() +
  coord_flip()
```



Neste *boxplot* do preço conseguimos notar imediatamente a existência de muitos valores atípicos, que equivalem a preços muito altos tendo em conta a média de preços dos registos que é de 303.465 USD. Estes preços vão sem dúvida interferir com as nossas análises. Estes preços conseguem ser explicados quando analisamos a sua fonte.

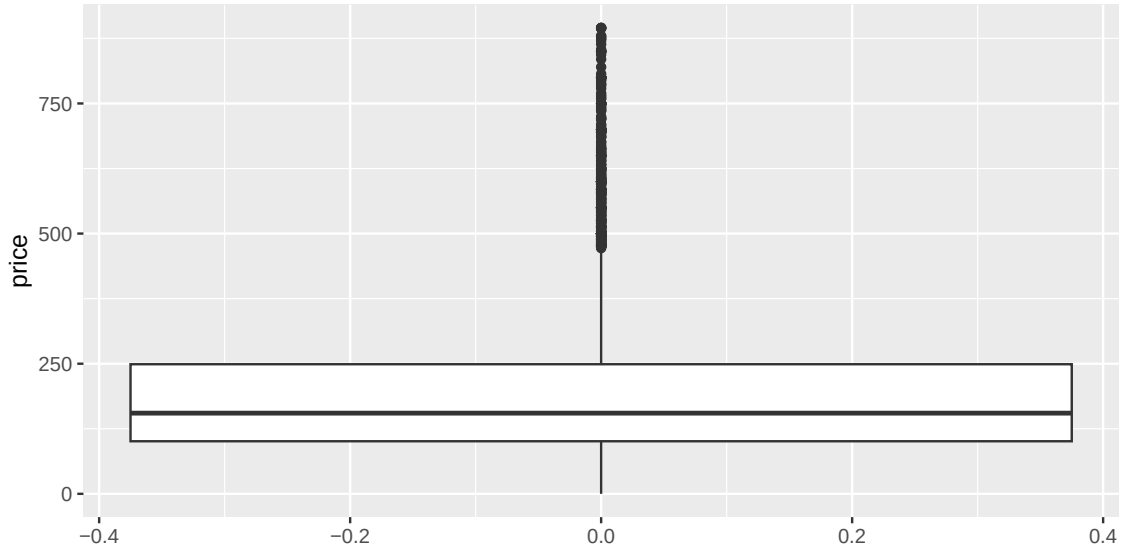
```
df %>% select(name, price) %>% arrange(-price) %>% head(7) %>% showT(TRUE)
```

name	price
Harbor Court Hotel, Bay View King Room	25000
Harbor Court Hotel, Bay View Queen Room	25000
Hotel Griffon by the Bay, queen bedded room	25000
1-Bedroom Suite with One Bed and One Sofabed at Fairmont San Francisco by Suiteness	20000
Suite plus Connecting Room with Three Beds and One Sofabed at Fairmont San Francisco by Suiteness	20000
Suite plus Connecting Room with Two Beds and One Sofabed at Fairmont San Francisco by Suiteness	20000
1-Bedroom Suite with One Bed at Fairmont San Francisco by Suiteness	20000

Assim, estes preços equivalem a alojamentos de luxo, que pela sua natureza terão de ser tratadas de forma diferente quando for feita a modelação, uma vez que não são comparáveis com o resto dos alojamentos.

Como tentativa de mitigar estes valores muito altos (*outliers*), provavelmente será necessário fazer uma transformação logarítmica do objetivo, de forma a reduzir a influência destes valores no modelo.

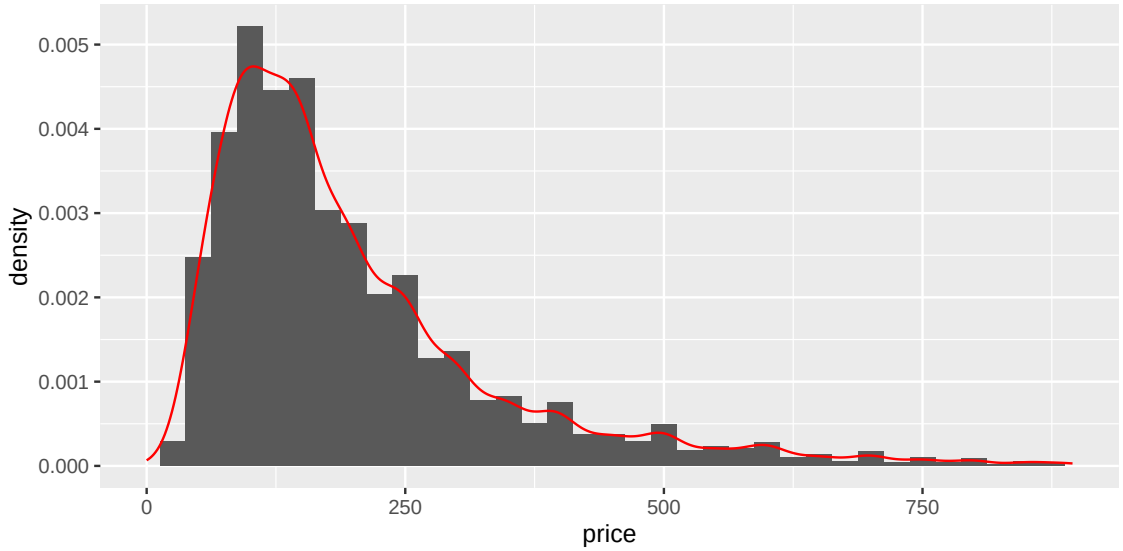
```
upper_limit <- quantile(df$price, 0.975) + 20
ggplot(data=df, aes(price)) +
  geom_boxplot() +
  coord_flip() +
  xlim(0,upper_limit)

## Warning: Removed 158 rows containing non-finite values (`stat_boxplot()`).
```



Este *boxplot* mostra que a maioria dos alojamentos tem preços entre 103 e 254 USD. Este facto torna-se ainda mais evidente quando analisamos a distribuição dos preços.

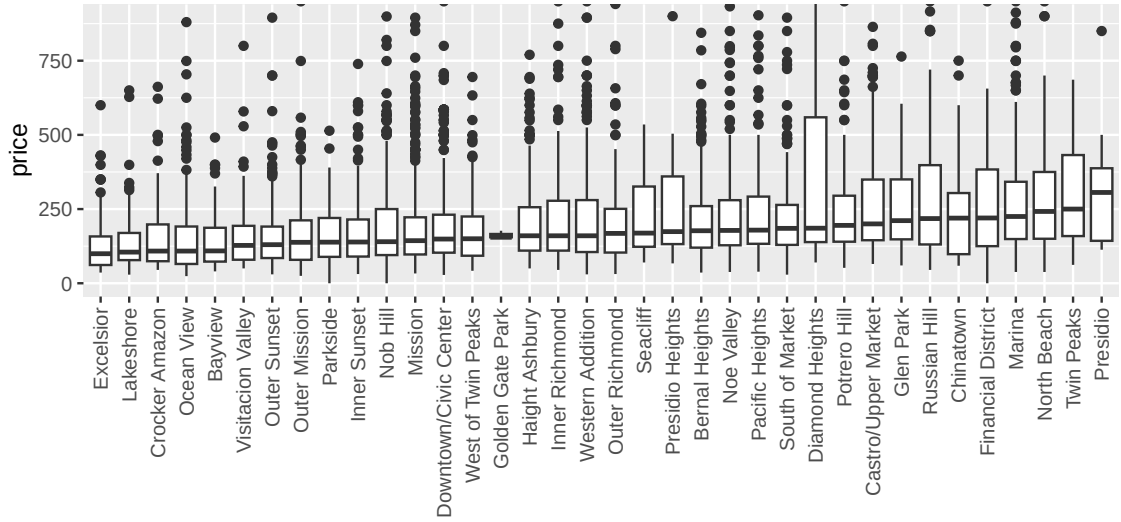
O gráfico mostra também que há muitos alojamentos fora destes limites, podendo ser valores atípicos também, embora não tão extremos como aqueles vistos anteriormente. No entanto, à primeira vista estes não devem ser valores atípicos, devido à sua quantidade, mesmo quando comparado com o número de registos.

```
ggplot(data=df, aes(price)) +
  geom_histogram(binwidth=25, aes(y = ..density..)) +
  geom_density(color="red") +
  xlim(0, upper_limit)
```



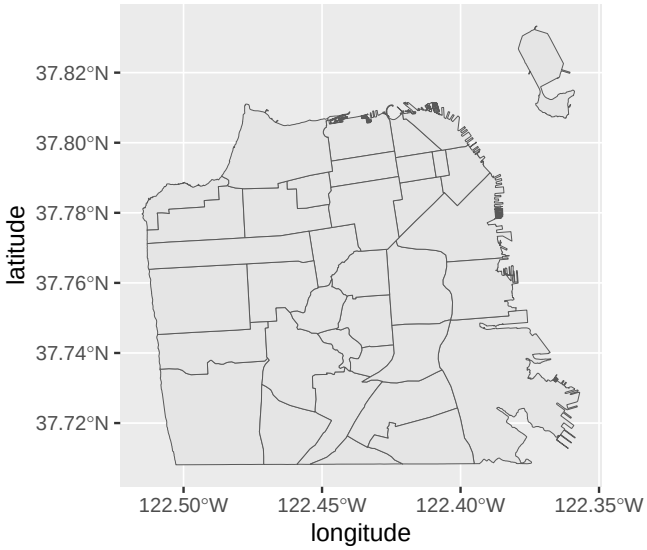
A distribuição de preços apresentada demonstra que a maioria dos alojamentos se encontram no limite mostrado anteriormente, e a distribuição parece aproximar-se de uma distribuição χ^2_k, com um pequeno grau de liberdade. Curiosamente, o gráfico mostra que os preços parecem aumentar algumas vezes a cada 50 USD, o que pode ser devido ao facto de que os hospedeiros escolhem preços redondos, como 50, 100, 175, etc. Este fenómeno parece ser mais visível nos 250 e nos 500.

```
df %>% ggplot(aes(y=price, x = forcats::fct_reorder(neighbourhood, price, .fun=median)))
```



Os *boxplots* mostram que os preços dos alojamentos não variam bastante de acordo com o bairro sendo que o ponto médio não varia bastante entre bairros, excepto os bairros *Twin Peaks*, e *Presidio*. No entanto, o gráfico mostra uma grande variância dos preços em todos os bairros, exceto no bairro Golden Gate Park.

```
shape_plot +
  stat_summary_hex(data = df, aes(x=longitude, y=latitude, z= log(price)), alpha=0.8) +
  scale_fill_viridis_c() +
  theme(legend.position = "right")

## Warning: Removed 3 rows containing non-finite values (`stat_summary_hex()`).
## Warning: Computation failed in `stat_summary_hex()`
## Caused by error in `compute_group()`:
## ! The package `hexbin` is required for `stat_summary_hex()`
```



Este gráfico demonstra a variariadade presente nos vários distritos, e da mesma forma não é possível descrever nenhum padrão *a priori* para os preços.

```
df %>%
  group_by(room_type) %>%
  summarise(
    n = n(),
    freq = n()/nrow(.),
    averagePrice = mean(price),
    sd = sd(price),
    min = min(price),
    max = max(price)
  ) %>% showT()
```

room_type	n	freq	averagePrice	sd	min	max
Entire home/apt	4243	0.6400664	275.8965	406.8473	33	12000
Hotel room	65	0.0098054	266.2154	211.0756	0	1220
Private room	2239	0.3377583	360.1474	1972.6562	0	25000
Shared room	82	0.0123699	211.7683	1101.4410	25	10000

A tabela mostra que a maioria dos alojamentos são apartamentos ou casas inteiras, enquanto que os quartos privados são menos frequentes. Mostra também a pequena quantidade de quartos de hotéis e de alojamentos partilhados, sendo estes apenas 2% dos registos. Isto pode levar a que seja necessário o uso de alguma técnica de *oversampling* para os quartos de hotel e para alojamentos partilhados, de forma a aumentar a quantidade de registos destes tipos de alojamentos.

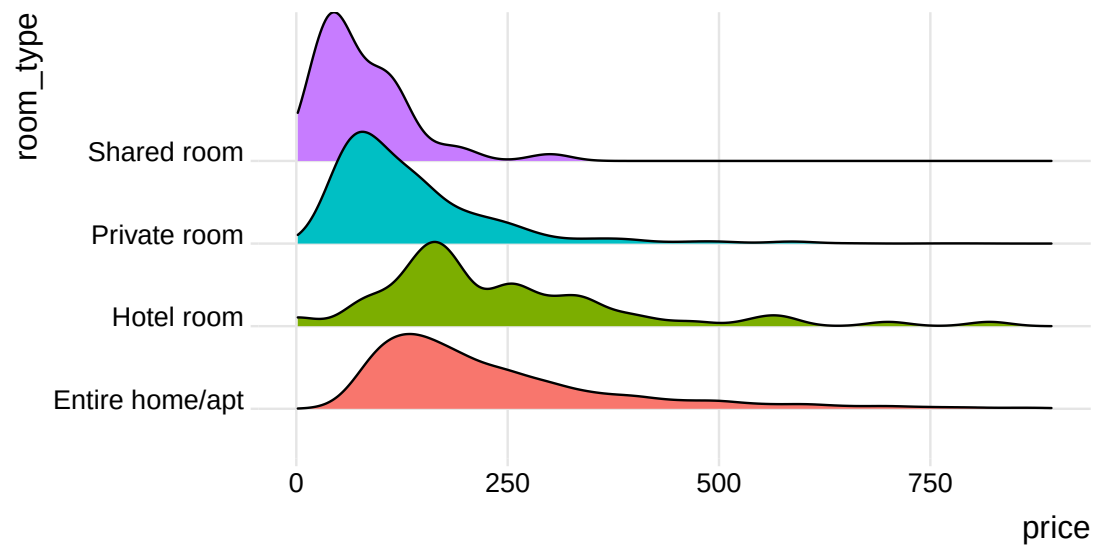
A tabela também expõe que os quartos privados são os mais caros em média, enquanto que os alojamentos partilhados são os mais baratos. Esta observação é esperada na parte que diz respeito aos quartos partilhados, no entanto é supreendente que os quartos privados sejam mais caros em média que as casas inteiras e os quartos de hotel. Isto pode ser porque a diferença entre ”quarto privado”e ”quarto de hotel”pode ser confusa, tanto que os hoteis de alto preço notados em cima estão caracterizados como ”quartos privados”. Isto pode explicar também o grande desvio padrão das variáveis.

```
df %>%
  group_by(room_type) %>%
  filter(price < upper_limit, price > 0) %>%
  summarise(
    n = n(),
    freq = n()/nrow(.),
    averagePrice = mean(price),
    sd = sd(price),
    min = min(price),
    max = max(price)
  ) %>% showT()
```

room_type	n	freq	averagePrice	sd	min	max
Entire home/apt	4131	0.6391769	233.9990	147.74938	33	880
Hotel room	61	0.0094383	248.9180	151.65776	72	820
Private room	2191	0.3390067	135.2793	95.85747	24	800
Shared room	80	0.0123782	78.3125	55.54006	25	300

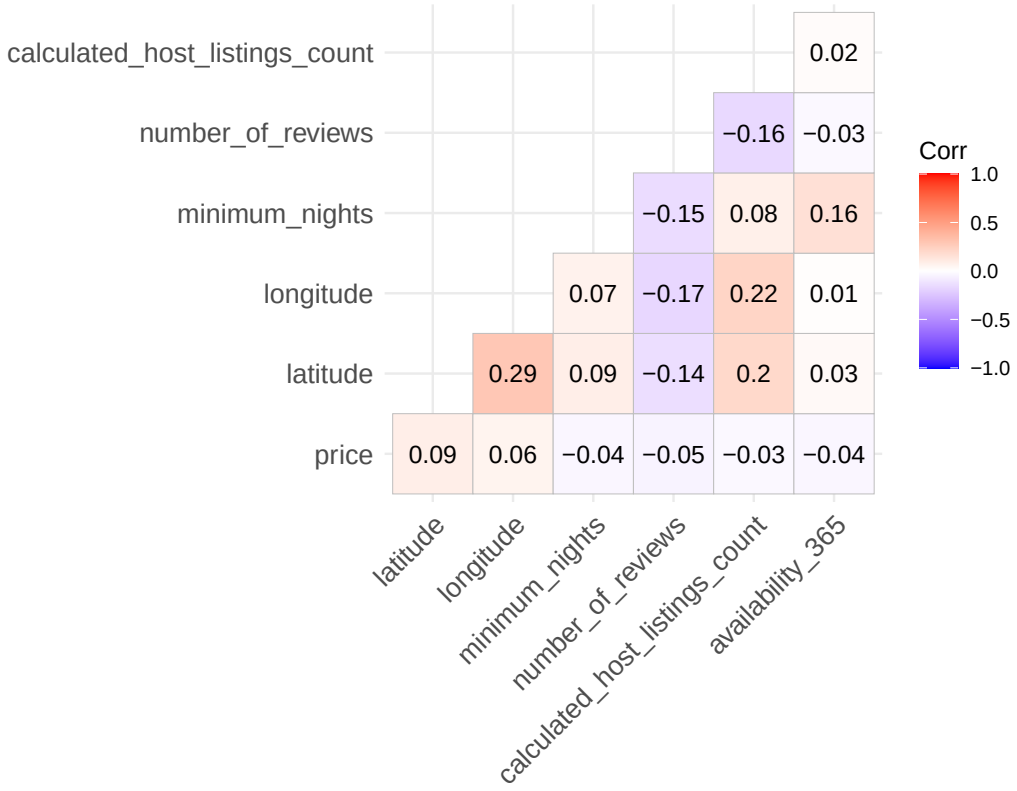
Se excluirmos os apartamentos de luxo, conseguimos observar valores mais esperados; quartos de hoteis serem os mais caros com preços aproximados aos das casas inteiras, e os alojamentos partilhados serem os mais baratos. Os preços médios dos quartos privados desceram significativamente. Isto pode ser explicado pela classificação de alojamentos de luxo como "quartos privados". Deste modo suspeitamos que, sem os quartos de luxo, esta nova categoria identifica-se mais com albergues.

```
ggplot(df, aes(x=price, y=room_type, fill=room_type)) +
  geom_density_ridges() + xlim(0, upper_limit) +
  theme_ridges() + theme(legend.position = "none")

## Picking joint bandwidth of 24.6
```



O diagrama apresentado demonstra as distribuições dos preços por tipo de alojamento. Este corrobora que os quartos de hóteis são os mais caros, mas parece demonstrar também que a maioria dos outros grupos se encontram com preços semelhantes, com uma diferença na densidade na cauda do gráfico, explicando assim a alta média de apartamentos inteiros. desta forma, a maioria dos alojamentos inteiros estão de acordo com quartos privados, mas existem mais alojamentos inteiros mais caros.

```
df %>%
select(price, latitude, longitude,minimum_nights, number_of_reviews, calculated_host_list
cor(use = "complete.obs") %>%
ggcorrplot(lab = TRUE, type = "lower")
```



O gráfico de correlações não mostra também nenhuma correlação significante entre o preço e as outras variáveis numéricas analisaveis. Mostra também uma pouca correlação entre as variáveis independentes.

```
recipeUpSample <- recipe(price ~ neighbourhood + latitude + longitude + room_type + minim
  step_upsample(room_type, skip=FALSE, over_ratio = 0.363) %>%
  step_dummy(neighbourhood, room_type) %>%
  prep()
recipeNoUpSample <- recipe(price ~ neighbourhood + latitude + longitude + room_type + min
step_dummy(neighbourhood, room_type) %>%
prep()
dfAll <- bake(recipeUpSample, df %>% filter(price > 0, neighbourhood != "Golden Gate Park
dfAll %>% group_by(isHotel = room_type_Hotel.room, isPrivateRoom = room_type_Private.room
```

isHotel	isPrivateRoom	isSharedRoom	n	freq	averagePrice
0	0	0	4242	0.4439096	280.2638
0	0	1	1539	0.1610506	224.6303
0	1	0	2236	0.2339891	405.3028
1	0	0	1539	0.1610506	268.8304

Os nossos preços nulos foram retirados pois são aparentes erros devido à natureza da variável, e os nossos registos do *Golden Gate Park* foram retirados pois são apenas 3, e não é possível fazer uma análise significativa com tão poucos registos.
A quantidade de hotéis e alojoamentos partilhados foram aumentada para 1539 (ainda inclui desequilíbrio, mas devido à realmente pequena quantidade destes decidimos não

equilibrar demasiado), e as nossas variáveis categóricas foram transformadas em *dummies*, de forma a tornar possível a sua análise.

Para analisar e comparar modelos, vai ser usado *in-sample* e analisado as estatísticas do coeficiente de determinação R^2, o *MAPE*, e o *rRMSE*. A estatística F vai ser insignificante pare o nosso modelo, devido à grande quantidade de observações, levando a uma rejeição da hipótese nula constante. Para comparar modelos, vai ser observado o *AIC* de cada, quando apropriado. Para verificar os pressupostos, vai ser analisado se a média dos resíduos padrozinados é praticamente zero, o teste de *Breusch-Pagan* para verificar a homocedasticidade, o teste de *Breusch-Godfrey* para verificar a autocorrelação, e o teste de *Jarque-Bera* para verificar a normalidade dos resíduos. Em alguns modelos vai ser também analisado o gráficos de resíduos.

Em cada um destes testes, se o valor p for menor que 0.05, então rejeitamos H_0. No teste *Breusch-Pagan*, H_0 é que os resíduos são homocedásticos, e no teste *Breusch-Godfrey*, H_0 é que os resíduos não são autocorrelacionados. No teste *Jarque-Bera*, H_0 é que os resíduos são normais.

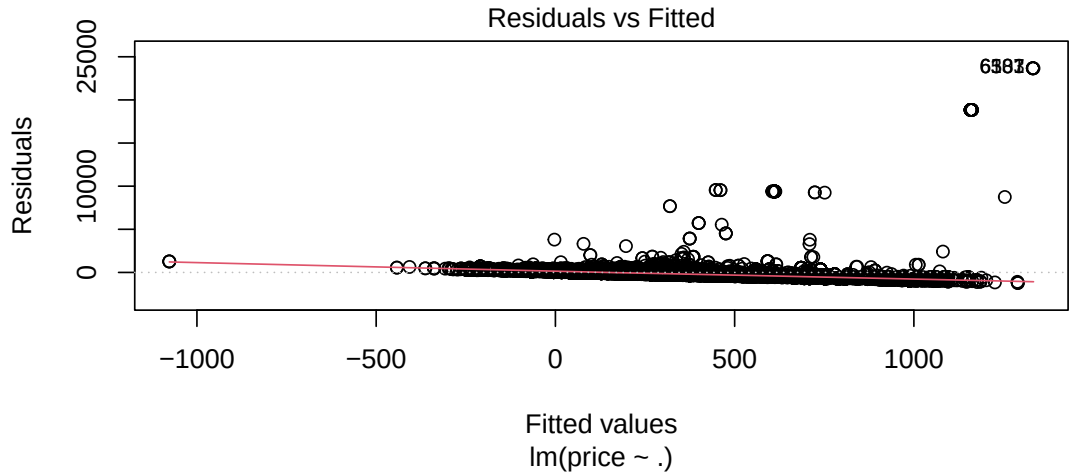
```
testModel <- function(fit, dfAlll, desc, transPrice = (\(x) x )){
  predicted <- predict(fit, dfAlll)
  fit %>% glance() %>% select(
    R2=r.squared,
    AIC=AIC) %>% mutate(
    R2 = formatC(R2, digits=3),
    AIC = formatC(AIC, digits=3),
    MAPE=mape(dfAlll$price %>% transPrice(), predicted) %>%
      format(digits=3),
    rRMSE=rrmse(dfAlll$price %>% transPrice(), predicted) %>%
      format(digits=3),
    "Breusch-Pagan"=validP(bptest(fit)$p.value, FALSE),
    "Breusch-Godfrey"=validP(bgtest(fit)$p.value, FALSE),
    "Jarque-Bera"=validP(jarque.bera.test(fit$residuals)$p.value, TRUE)
  ) %>% mutate(Model=desc) %>% relocate(Model)
}
```

O nosso primeiro modelo linear foi um modelo que diretamente relaciona as variáveis todas com o preço.

```
fit0 <- lm(price ~ ., dfAll)
testModel(fit0, dfAll, "fit0(price ~ .)") %>% t() %>% showT()
```

Model	fit0(price ~ .)
R2	0.0462
AIC	1.62e+05
MAPE	1.87
rRMSE	0.0402
Breusch-Pagan	p = 9.9e-58
Breusch-Godfrey	p > 0.05
Jarque-Bera	p < 0.05

```
plot(fit0, 1)
```



O primeiro modelo notavelmente não é apropriado, devido ao valor gigante de *MAPE* e ao pequeno valor de R^2. Ao observar o gráfico, e com a rejeição da hipótese de homocedasticidade, podemos concluir que uma transformação será necessária.

```
fit0.cube <- lm(price^3 ~ ., dfAll)
fit0.squared <- lm(price^2 ~ ., dfAll)
fit0.root <- lm(sqrt(price) ~ ., dfAll)
fit0.log <- lm(log(price) ~ ., dfAll)
fit0.invSqrt <- lm(1/sqrt(price) ~ ., dfAll)
fit0.inv <- lm(1/price ~ ., dfAll)
fit0.invSquared <- lm(1/price^2 ~ ., dfAll)
fit0.invCube <- lm(1/price^3 ~ ., dfAll)
rbind(
  testModel(fit0.cube, dfAll, "cube", (\(x) x^3 )),
  testModel(fit0.squared, dfAll, "squared", (\(x) x^2 )),
  testModel(fit0.root, dfAll, "root", sqrt),
  testModel(fit0.log, dfAll, "log", log),
  testModel(fit0.invSqrt, dfAll, "invSqrt", (\(x) 1/sqrt(x))),
  testModel(fit0.inv, dfAll, "inv", (\(x) 1/x)),
  testModel(fit0.invSquared, dfAll, "invSqdr", (\(x) 1/x^2)),
  testModel(fit0.invCube, dfAll, "invCube", (\(x) 1/x^3))
  ) %>% t() %>% showT(TRUE)
```

Model	cube	squared	root	log	invSqrt	inv	invSqdr	invCube
R2	0.0381	0.0395	0.123	0.366	0.521	0.585	0.594	0.55
AIC	5.42e+05	3.51e+05	7.02e+04	1.99e+04	-4.41e+04	-7.54e+04	-1.41e+05	-2.05e+05
MAPE	145302	355	0.291	0.0906	0.306	1.59	217	65635
rRMSE	0.191	0.153	0.00693	0.00138	0.00282	0.00546	0.012	0.0194
Breusch-Pagan	p = 4.27e-48	p = 1.74e-52	p = 2.2e-63	p = 8.77e-103	p = 1.34e-233	p = 5.87e-310	p = 7.91e-322	p = 1.13e-280
Breusch-Godfrey	p > 0.05	p > 0.05	p > 0.05	p > 0.05	p > 0.05	p > 0.05	p > 0.05	p > 0.05
Jarque-Bera	p < 0.05	p < 0.05	p < 0.05	p < 0.05	p < 0.05	p < 0.05	p < 0.05	p < 0.05

Comparando as estatísticas das várias transformações do preço, podemos concluir que a transformação dos modelos $log(y)$ e $\sqrt{y^{-1}}$ são as mais apropriadas, a primeira com um valor de *MAPE* e rRMSE mais baixo, e a segunda com um valor de R^2 mais alto. Nenhum desses modelos satifaz o pressuposto de homocedasticidade. Para continuará vai ser revisto os valores dos modelos em *out-of-sample*. Para isso a base de dados foi dividida em 80% para treino e 20% para teste.

```
set.seed(123)
dfSplit <- initial_split(df %>% filter(price > 0, neighbourhood != "Golden Gate Park"), p
dfTrain <- training(dfSplit) %>% bake(recipeUpSample, .)
dfTest <- testing(dfSplit) %>% bake(recipeNoUpSample, .)
```

O conjunto de treino usou o mesmo método de *oversampling* usado anteriormente, mas o conjunto de teste não foi alterado.

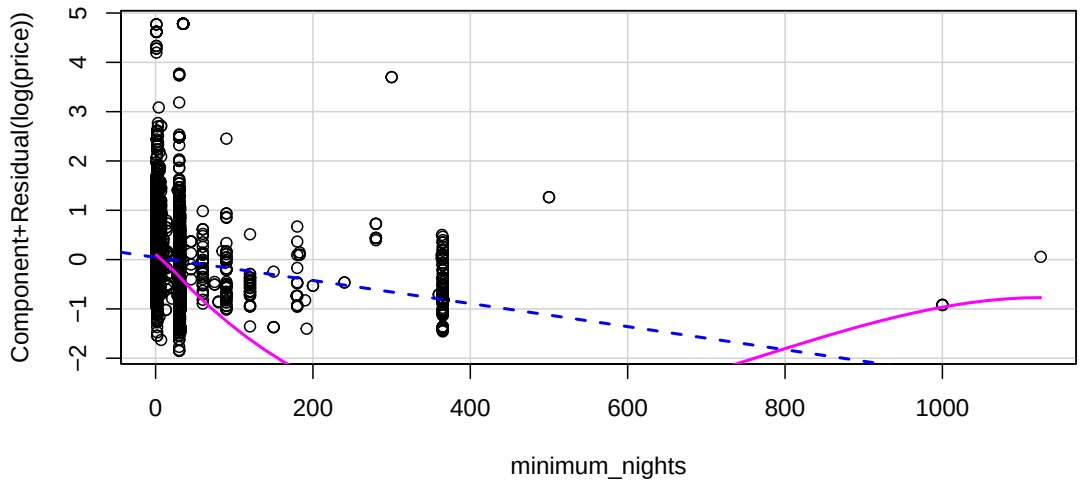
```
fit0.log.outOfSample <- lm(log(price) ~ ., dfTrain)
fit0.invSqrt.outOfSample <- lm(1/sqrt(price) ~ ., dfTrain)
rbind(
  testModel(fit0.log, dfAll, "log", log),
  testModel(fit0.invSqrt, dfAll, "invSqrt", (\(x) 1/sqrt(x))),
  testModel(fit0.log.outOfSample, dfTest, "logOut", log),
  testModel(fit0.invSqrt.outOfSample, dfTest, "invSqrtOut", (\(x) 1/sqrt(x)))
) %>% t() %>% showT(F)
```

Model	log	invSqrt	logOut	invSqrtOut
R2	0.366	0.521	0.392	0.525
AIC	1.99e+04	-4.41e+04	1.63e+04	-3.82e+04
MAPE	0.0906	0.306	0.0928	0.288
rRMSE	0.00138	0.00282	0.00351	0.0076
Breusch-Pagan	p = 8.77e-103	p = 1.34e-233	p = 6.24e-106	p = 1.05e-247
Breusch-Godfrey	p > 0.05	p > 0.05	p > 0.05	p > 0.05
Jarque-Bera	p < 0.05	p < 0.05	p < 0.05	p < 0.05

Após análise destes novos modelos, como o *MAPE* e o *rRMSE* do modelo com a transformação logarítmica é menor em ambos os conjuntos, considerámos esse modelo como melhor.

Como o pressuposto de homocedasticidade ainda não se verifica, fomos analisar os gráficos de resíduos parcial, de forma a verificar alguma não-linearidade nos dados.

```r
if(FALSE){
  crPlots(fit0.log)
}
```



Ao analisar os gráficos, apenas o gráfico dos resíduos parcial da variável *minimum_nights* mostra uma não linearidade. Como a variável mostra alguma significância no nosso modelo ($p < 0.05$ do teste t), vamos transformar essa variável e analisar e comparar o modelo resultado dessa.

```r
fit1.noMN <- lm(log(price) ~ . - minimum_nights, dfAll)
fit1 <- lm(log(price) ~ ., dfAll)
fit1.MNsq <- lm(log(price) ~ ., dfAll %>% mutate(mnsqd = minimum_nights^2, mncube = minim
rbind(
  testModel(fit1.noMN, dfAll, "noMN", log),
  testModel(fit1, dfAll, "withMN", log),
  testModel(fit1.MNsq, dfAll %>% mutate(mnsqd = minimum_nights^2, mncube = minimum_nights
) %>% t() %>% showT(F)
```

Model	noMN	withMN	MNsq
R2	0.348	0.366	0.406
AIC	2.01e+04	1.99e+04	1.92e+04
MAPE	0.0931	0.0906	0.0857
rRMSE	0.0014	0.00138	0.00133
Breusch-Pagan	p = 1.01e-109	p = 8.77e-103	p = 2.32e-103
Breusch-Godfrey	p > 0.05	p > 0.05	p > 0.05
Jarque-Bera	p < 0.05	p < 0.05	p < 0.05

Ao retirar a variável, os pressupostos mantêm-se, e as estatísticas de comparação pioram, como esperado. Infelizmente, estes presupostos mantém-se também mesmo com as novas variáveis. Como o *AIC* melhorou para o modelo "MNsq", este continuará a ser usado.

De forma a resolver a heterocedasticidade, vamos usar o método de *weighted least squares* (WLS), que é um método de regressão linear que usa pesos para cada observação, de forma a minimizar a soma dos quadrados dos resíduos.

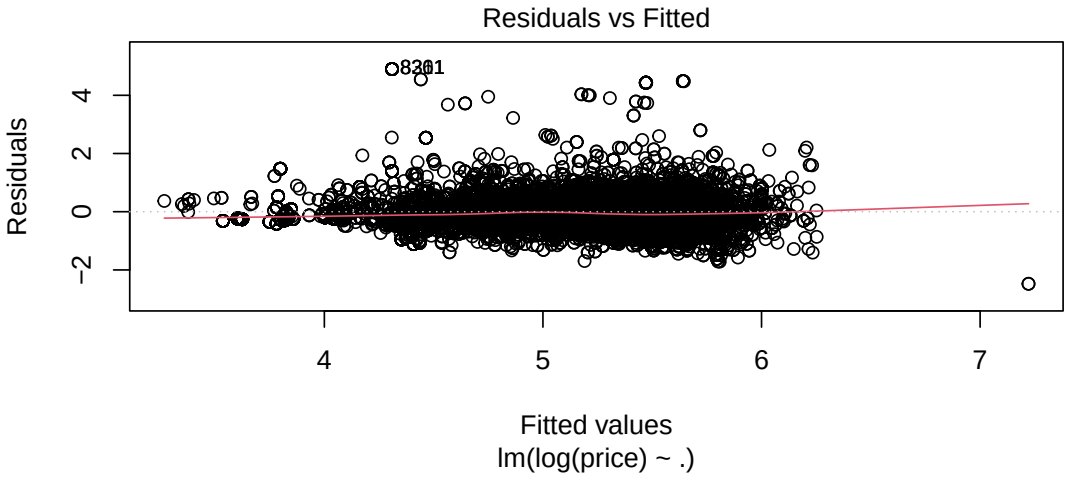
```
dfAllMN <- dfAll %>% mutate(mnsqd = minimum_nights^2, mncube = minimum_nights^3)
pfit1 <- lm(log(price) ~ ., dfAllMN)$fitted.values
fit2 <- lm(log(price) ~ ., dfAllMN)
fit2.sqdrN <- lm(log(price) ~ ., dfAllMN, weights=1/((1:nrow(dfAll))^2))
fit2.rootN <- lm(log(price) ~ ., dfAllMN, weights=1/((1:nrow(dfAll))^0.5))
fit2.rootPred <- lm(log(price) ~ ., dfAllMN, weights=(1/pfit1^0.5))
fit2.sqdrPred <- lm(log(price) ~ ., dfAllMN, weights=(1/pfit1^2))
rbind(
  testModel(fit2, dfAllMN, "noW", log),
  testModel(fit2.sqdrN, dfAllMN, "sqdrN", log),
  testModel(fit2.rootN, dfAllMN, "rootN", log),
  testModel(fit2.rootPred, dfAllMN, "rootPred", log),
  testModel(fit2.sqdrPred, dfAllMN, "sqdrPred", log)
) %>% t() %>% showT(F)
```

Model	noW	sqdrN	rootN	rootPred	sqdrPred
R2	0.406	0.958	0.367	0.415	0.44
AIC	1.92e+04	5.88e+04	2e+04	1.93e+04	1.96e+04
MAPE	0.0857	0.123	0.0873	0.0857	0.0861
rRMSE	0.00133	0.00178	0.00134	0.00133	0.00134
Breusch-Pagan	p = 2.32e-103	p > 0.05	p > 0.05	p = 6.32e-53	p > 0.05
Breusch-Godfrey	p > 0.05	p > 0.05	p > 0.05	p > 0.05	p > 0.05
Jarque-Bera	p < 0.05	p < 0.05	p < 0.05	p < 0.05	p < 0.05

Os modelos apresentados conseguiram não rejeitar a hipótese de homocedasticidade, embora as estatísticas de *MAPE* e *rRMSE* tenham aumentado levemente. Contudo, iremos usar um dos modelos com pesos pois estes assumem todos os pressupostos analisados. Baseado no *AIC*, o modelo "sqdrPred"é o melhor.

De forma a confirmar a nossa escolha, vamos analisar os resíduos do modelo.

```
plot(fit2.sqdrPred, 1)
```



O gráfico de resíduos não mostra tendência, nem heterocedasticidade evidente, logo podemos concluir que o modelo pode ser adequado para prever o preço. De forma a assegurar a afirmação, vai ser feita também o teste *out-of-sample* com estes últimos modelos.

```
dfTrainMN <- dfTrain %>%
  mutate(mnsqd = minimum_nights^2, mncube = minimum_nights^3)
dfTestMN <- dfTest %>%
  mutate(mnsqd = minimum_nights^2, mncube = minimum_nights^3)
pfit1.outOfSample <- lm(log(price) ~ ., dfTrainMN)$fitted.values
fit2.sqdrN.outOfSample <- lm(log(price) ~ ., dfTrainMN,
  weights=1/((1:nrow(dfTrain))^2))
fit2.rootN.outOfSample <- lm(log(price) ~ ., dfTrainMN,
  weights=1/((1:nrow(dfTrain))^0.5))
fit2.sqdrPred.outOfSample <- lm(log(price) ~ ., dfTrainMN,
  weights=(1/pfit1.outOfSample^2))
rbind(
  testModel(fit2.sqdrN, dfAllMN, "sqdrN", log),
  testModel(fit2.sqdrN.outOfSample, dfTestMN, "sqdrNOut", log),
  testModel(fit2.rootN, dfAllMN, "rootN", log),
  testModel(fit2.rootN.outOfSample, dfTestMN, "rootNOut", log),
  testModel(fit2.sqdrPred, dfAllMN, "sqdrPred", log),
  testModel(fit2.sqdrPred.outOfSample, dfTestMN, "sqdrPredOut", log)
) %>% t() %>% showT(F)
```

Model	sqdrN	sqdrNOut	rootN	rootNOut	sqdrPred	sqdrPredOut
R2	0.958	0.914	0.367	0.385	0.44	0.486
AIC	5.88e+04	5.33e+04	2e+04	1.68e+04	1.96e+04	1.56e+04
MAPE	0.123	0.106	0.0873	0.0848	0.0861	0.0869
rRMSE	0.00178	0.00394	0.00134	0.0033	0.00134	0.00337
Breusch-Pagan	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$
Breusch-Godfrey	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$	$p > 0.05$
Jarque-Bera	$p < 0.05$	$p < 0.05$	$p < 0.05$	$p < 0.05$	$p < 0.05$	$p < 0.05$

Como previsto, os modelos mantém aproximadamente as suas estatísticas de *MAPE* e *rRMSE*, mas estas aumentam. O modelo "sqdrPred"mantém se como o melhor modelo

estiamdo. Os coeficientes do modelo são apresentados na tabela seguinte.

```
options(digits=3)
fit<- lm(log(price) ~ ., dfAllMN, weights=1/((1:nrow(dfAll))^0.5))
fit %>% tidy() %>% mutate(estimate = format(estimate, scientific=T), p.value = format(p.value, scientific=T))
```

term	estimate	std.error	statistic	p.value
(Intercept)	3.47e+02	140.726	2.465	1.37e-02
latitude	9.79e+00	1.336	7.331	2.48e-13
longitude	5.81e+00	1.000	5.807	6.55e-09
minimum_nights	-1.70e-02	0.001	-32.490	1.12e-219
number_of_reviews	-1.35e-03	0.000	-17.971	4.86e-71
calculated_host_listings_count	-3.42e-03	0.000	-14.917	9.26e-50
availability_365	2.74e-05	0.000	0.524	6.01e-01
neighbourhood_Bernal.Heights	3.33e-01	0.071	4.721	2.38e-06
neighbourhood_Castro.Upper.Market	6.06e-01	0.092	6.563	5.53e-11
neighbourhood_Chinatown	-3.44e-01	0.114	-3.024	2.50e-03
neighbourhood_Crocker.Amazon	4.59e-01	0.103	4.435	9.33e-06
neighbourhood_Diamond.Heights	7.24e-01	0.293	2.473	1.34e-02
neighbourhood_Downtown.Civic.Center	-1.36e-01	0.100	-1.362	1.73e-01
neighbourhood_Excelsior	1.96e-01	0.081	2.420	1.55e-02
neighbourhood_Financial.District	1.94e-01	0.108	1.803	7.15e-02
neighbourhood_Glen.Park	4.98e-01	0.108	4.629	3.73e-06
neighbourhood_Haight.Ashbury	4.99e-01	0.104	4.793	1.67e-06
neighbourhood_Inner.Richmond	2.41e-01	0.129	1.868	6.18e-02
neighbourhood_Inner.Sunset	3.18e-01	0.117	2.717	6.59e-03
neighbourhood_Lakeshore	5.87e-01	0.119	4.925	8.56e-07
neighbourhood_Marina	1.09e-01	0.129	0.843	4.00e-01
neighbourhood_Mission	1.82e-01	0.078	2.334	1.96e-02
neighbourhood_Nob.Hill	1.97e-01	0.108	1.827	6.77e-02
neighbourhood_Noe.Valley	4.96e-01	0.084	5.914	3.46e-09
neighbourhood_North.Beach	1.99e-01	0.123	1.619	1.06e-01
neighbourhood_Ocean.View	4.57e-01	0.097	4.730	2.27e-06
neighbourhood_Outer.Mission	3.27e-01	0.089	3.690	2.26e-04
neighbourhood_Outer.Richmond	3.38e-01	0.147	2.305	2.12e-02
neighbourhood_Outer.Sunset	4.52e-01	0.130	3.480	5.04e-04
neighbourhood_Pacific.Heights	2.76e-01	0.122	2.268	2.33e-02
neighbourhood_Parkside	4.61e-01	0.123	3.743	1.83e-04
neighbourhood_Potrero.Hill	2.89e-01	0.082	3.543	3.97e-04
neighbourhood_Presidio	2.44e-01	0.262	0.930	3.52e-01
neighbourhood_Presidio.Heights	1.06e-01	0.148	0.716	4.74e-01
neighbourhood_Russian.Hill	8.11e-02	0.126	0.642	5.21e-01
neighbourhood_Seacliff	6.01e-01	0.343	1.753	7.96e-02
neighbourhood_South.of.Market	5.36e-02	0.090	0.595	5.52e-01
neighbourhood_Twin.Peaks	6.55e-01	0.119	5.522	3.45e-08
neighbourhood_Visitacion.Valley	3.04e-01	0.098	3.107	1.90e-03
neighbourhood_West.of.Twin.Peaks	3.40e-01	0.103	3.286	1.02e-03
neighbourhood_Western.Addition	1.83e-01	0.104	1.758	7.88e-02
room_type_Hotel.room	-2.72e-01	0.025	-10.982	6.87e-28
room_type_Private.room	-5.49e-01	0.020	-27.950	2.80e-165
room_type_Shared.room	-1.14e+00	0.027	-42.328	0.00e+00
mnsqd	5.35e-05	0.000	26.809	1.09e-152
mncube	-3.51e-08	0.000	-23.967	2.63e-123

Referências

[1] tinyurl.com/DataDictAirbnb. Accessed: 24/11/2022.

[2] About Inside Airbnb. insideairbnb.com/about.html. Acessado: 24/11/2022.