Assignment 8

Veronika Palotai

11/10/2019

Exercise 1.

Import Libraries

```r
library(ggplot2)
library(tidyverse)
library(scales)
library(ggstance)
library(modelr)
library(gridExtra)
library(ggpubr)
library(data.table)
```

- Import from csv

Setting the Path

```r
dir <-  "D:/Egyetem/CEU/Coding_1/R-Coding/"
```

Location Folders

```r
data_in <- paste0(dir,"da_data_repo/hotels-europe/clean/")
```

Loading Datasets

```r
hotels_prices <- read_csv(paste0(data_in,"hotels-europe_price.csv"))
hotels_features <- read_csv(paste0(data_in,"hotels-europe_features.csv"))
```

Joining the two datasets

```r
hotels <- hotels_features %>%
  full_join(hotels_prices, by = "hotel_id")
```

Using anti-join to see which hotels do not have matches in the price data

```r
hotels_features %>%
  anti_join(hotels_prices, by = "hotel_id")
```

Looks like there is only one such row, the first which actually contains NA values only.

Exercise 2.

- **Filter to Vienna**

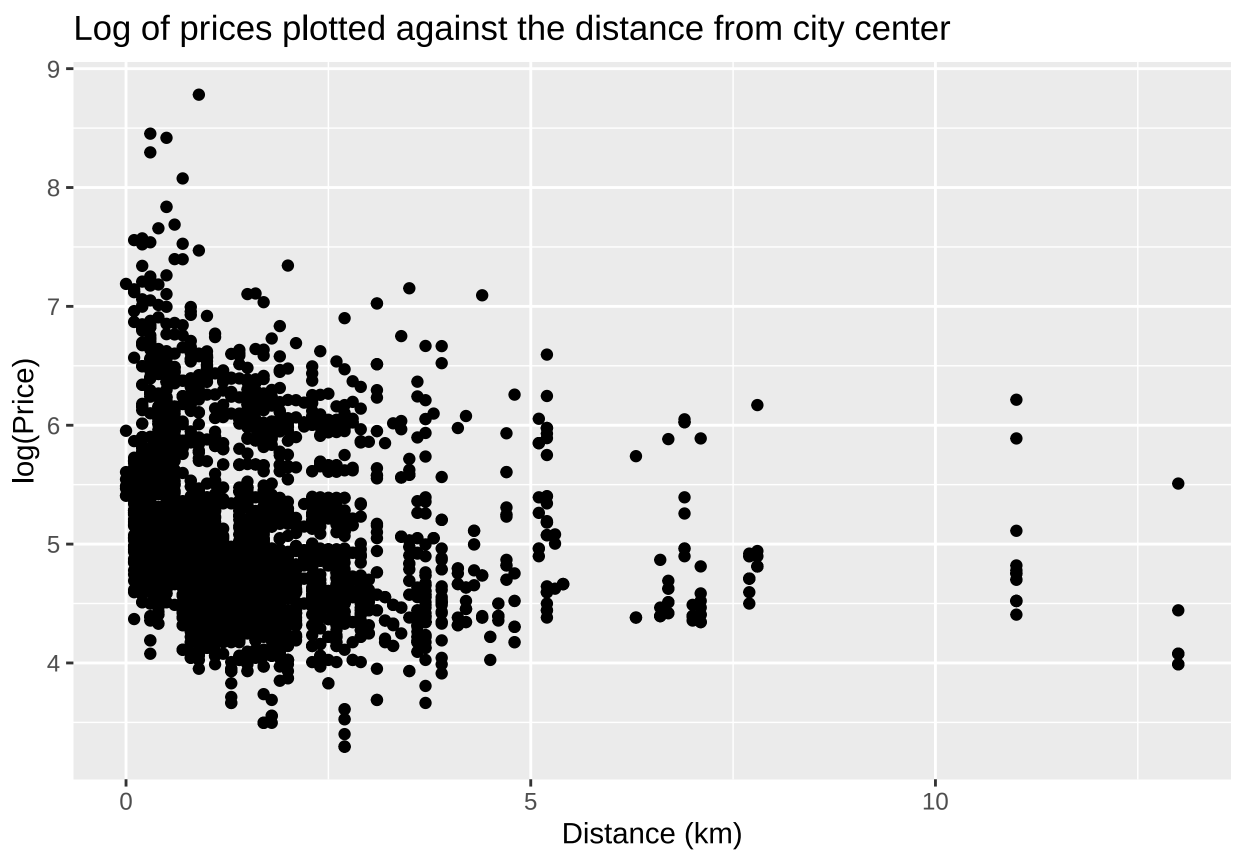
```r
vienna <- hotels %>%
  filter(city == 'Vienna')
```

- **Pick variables and models**

1. How prices and distance from the city center relate

Let's plot the natural logarithm of the prices against the distance from the city center! Taking the logarithm is necessary since a linear model was not suitable for the graph as it was.

```r
price_dist <- vienna %>%
  transmute(price, distance)

price_dist$price <- log(price_dist$price)

price_dist %>%
  na.exclude() %>%
  ggplot(mapping = aes(x = price_dist$distance, y = price_dist$price)) +
  geom_point() +
  labs(x = "Distance (km)",
       y = "log(Price)",
       title = "Log of prices plotted against the distance from city center")
```

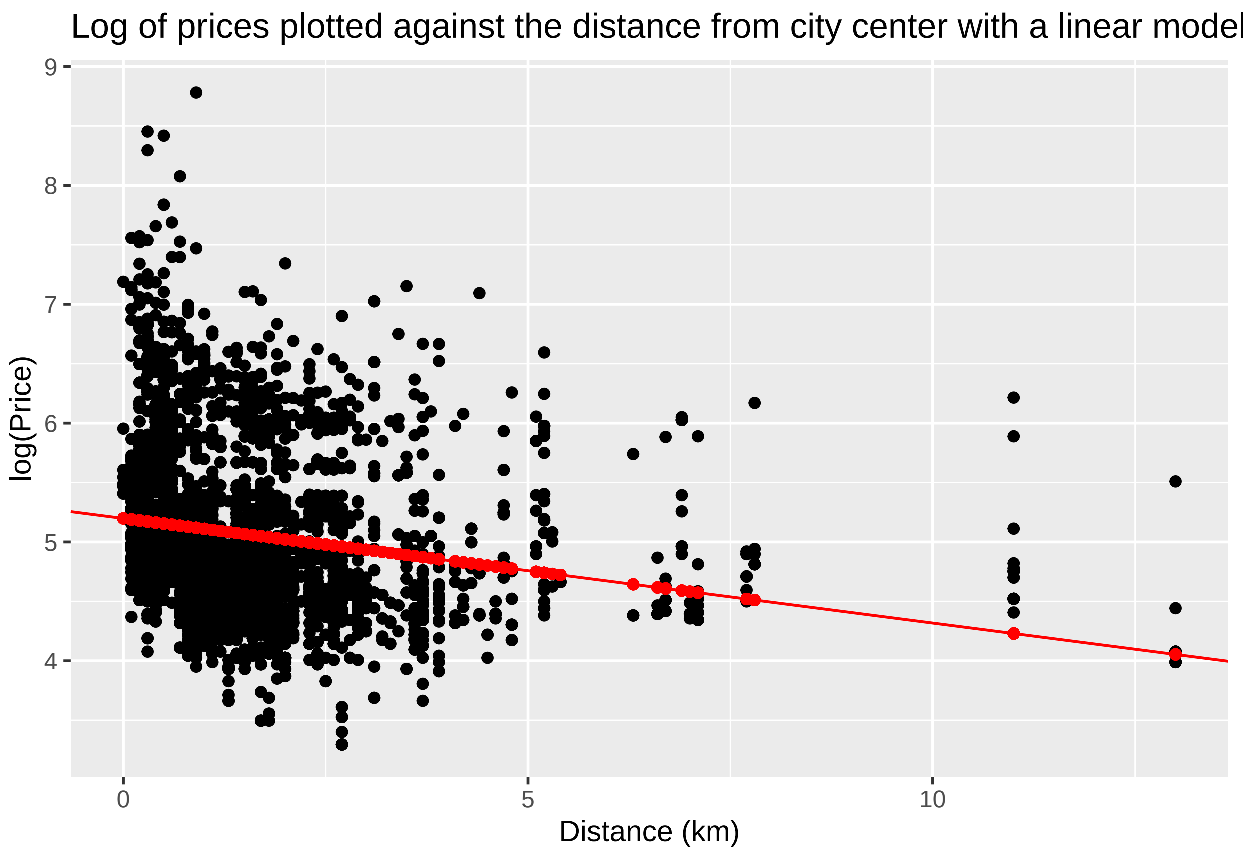


Model

```
mod1 <- lm(price ~ distance, data = price_dist)

coef(mod1)
```

```
## (Intercept)    distance
##  5.19812681 -0.08804268
```

```
df <- price_dist %>%
  add_predictions(mod1) %>%
  add_residuals(mod1)

ggplot(df, aes(df$distance,df$price)) +
  geom_point() +
  geom_point(aes(df$distance,df$pred), color = "red") +
  geom_abline(aes(intercept = coef(mod1)[1], slope = coef(mod1)[2]), color = "red") +
  labs(x = "Distance (km)",
       y = "log(Price)",
       title = "Log of prices plotted against the distance from city center with a linear model")
```

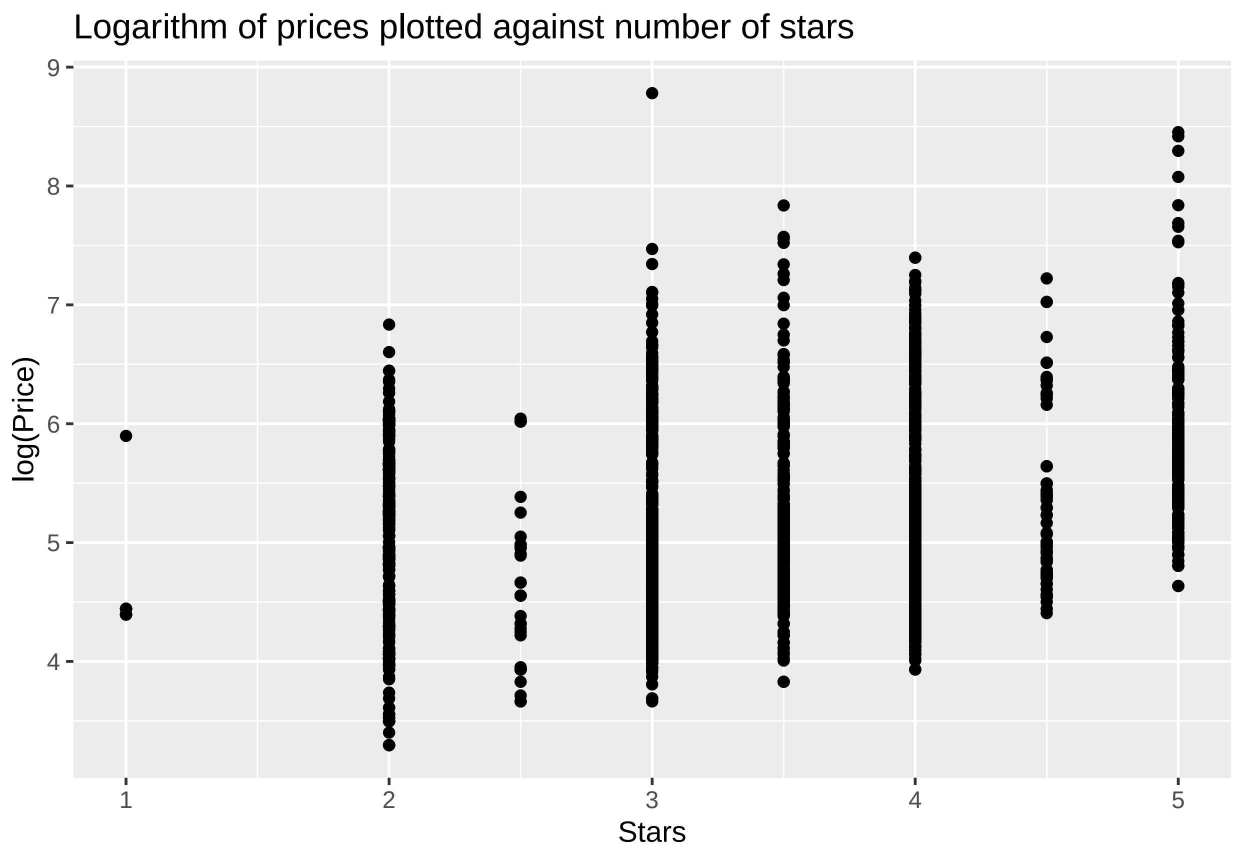


Conclusions: `ln(Price)^E = 5.19812681 - 0.08804268*Distance`. This is a log-level type regression, 5.19812681 is the average ‘ln(Price)’ when the ‘Distance’ is zero. The much more meaningful information is, however, that ‘Price’ is 8.804268% lower on average for observations having one unit higher ‘Distance’.

2. How prices and stars relate

Let's plot the natural logarithm of prices against the number of stars!

```r
price_stars <- vienna %>%
  transmute(price, stars)

price_stars$price <- log(price_stars$price)

price_stars %>%
  na.exclude() %>%
  ggplot(aes(x = stars, y = price)) +
  geom_point() +
  labs(x = "Stars",
       y = "log(Price)",
       title = "Logarithm of prices plotted against number of stars")
```

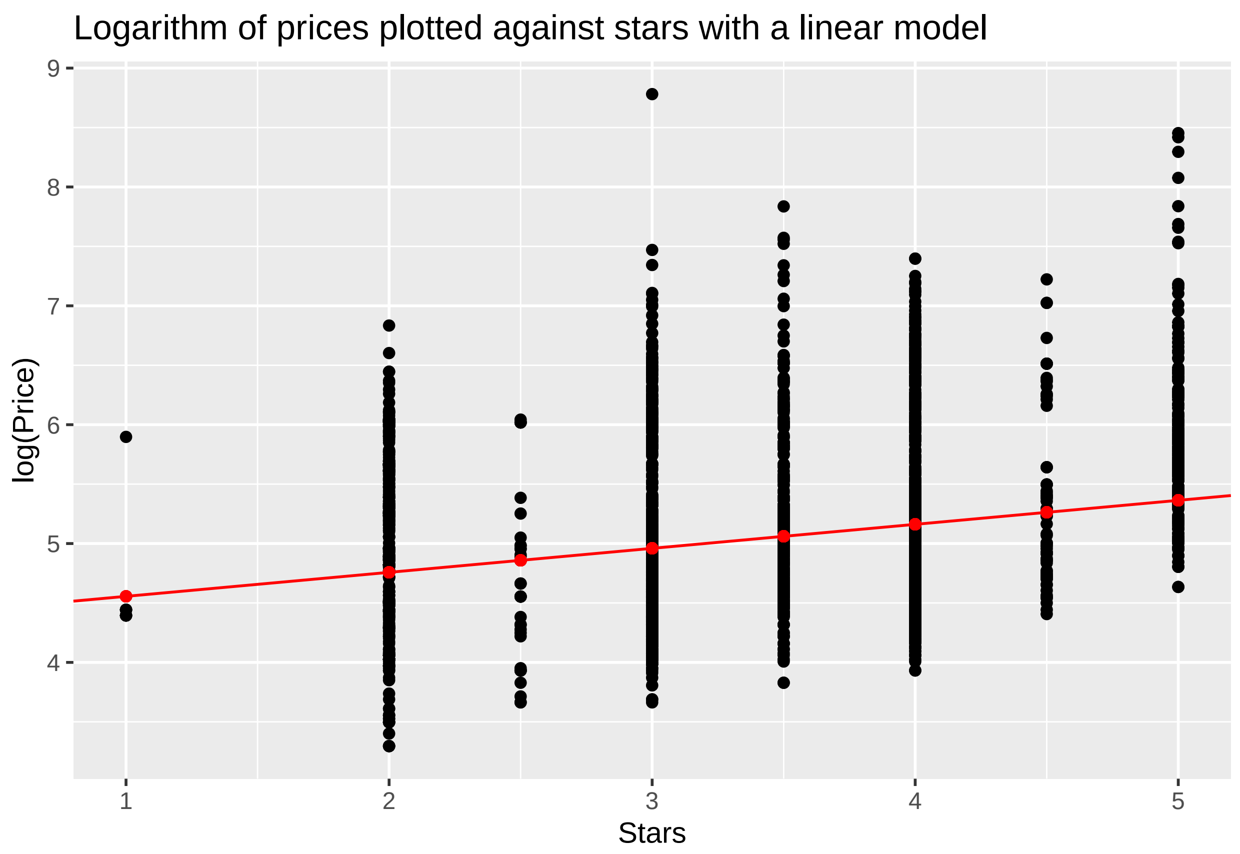


Model

```r
mod2 <- lm(price ~ stars, data = price_stars)

coef(mod2)
```

```
## (Intercept)       stars
##   4.3537876   0.2019431
```

```
df2 <- price_stars %>%
  add_predictions(mod2) %>%
  add_residuals(mod2)

ggplot(df2, aes(df2$stars,df2$price)) +
  geom_point() +
  geom_point(aes(df2$stars,df2$pred), color = "red") +
  geom_abline(aes(intercept = coef(mod2)[1], slope = coef(mod2)[2]), color = "red") +
  labs(x = "Stars",
       y = "log(Price)",
       title = "Logarithm of prices plotted against stars with a linear model")
```



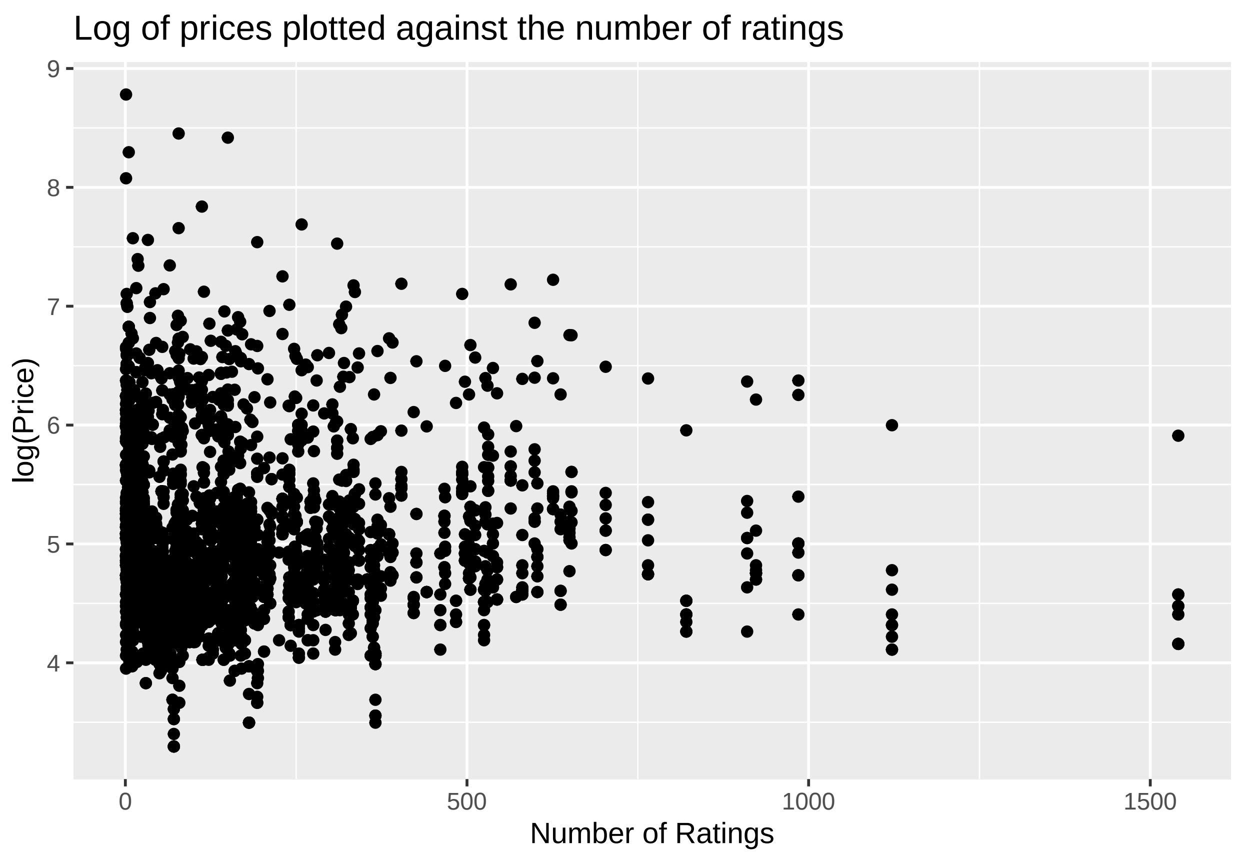
Conclusions: `ln(Price)^E = 4.3537876 + 0.2019431*Stars`. This is a log-level type regression, 4.3537876 is the average 'ln(Price)' when 'Stars' is zero. The much more meaningful information is, however, that 'Price' is 20.19431% higher on average for observations having one unit higher 'Stars'.

3. How prices and the number of ratings relate

Let's plot the natural logarithm of prices against the number of ratings!

```
price_rating <- vienna %>%
  transmute(price, rating_reviewcount)

price_rating$price <- log(price_rating$price)
```

```r
price_rating %>%
  na.exclude() %>%
  ggplot(aes(x = rating_reviewcount, y = price)) +
  geom_point() +
  labs(x = "Number of Ratings",
       y = "log(Price)",
       title = "Log of prices plotted against the number of ratings")
```

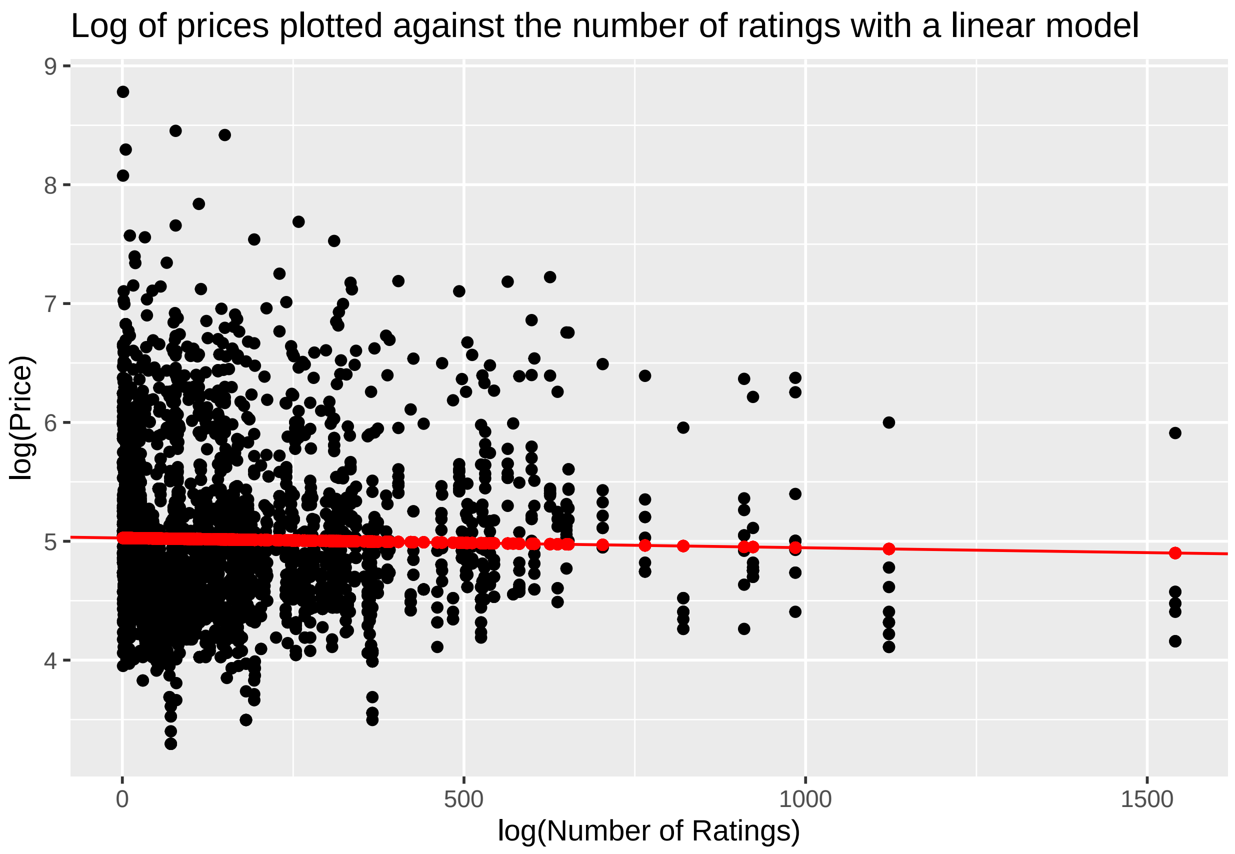


Model

```r
mod3 <- lm(price ~ rating_reviewcount, data = price_rating)

coef(mod3)
```

```
##        (Intercept) rating_reviewcount
##       5.0277153307      -0.0000822596
```

```r
df3 <- price_rating %>%
  add_predictions(mod3) %>%
  add_residuals(mod3)

ggplot(df3, aes(df3$rating_reviewcount,df3$price)) +
  geom_point() +
  geom_point(aes(df3$rating_reviewcount,df3$pred), color = "red") +
  geom_abline(aes(intercept = coef(mod3)[1], slope = coef(mod3)[2]), color = "red") +
  labs(x = "log(Number of Ratings)",
       y = "log(Price)",
       title = "Log of prices plotted against the number of ratings with a linear model")
```



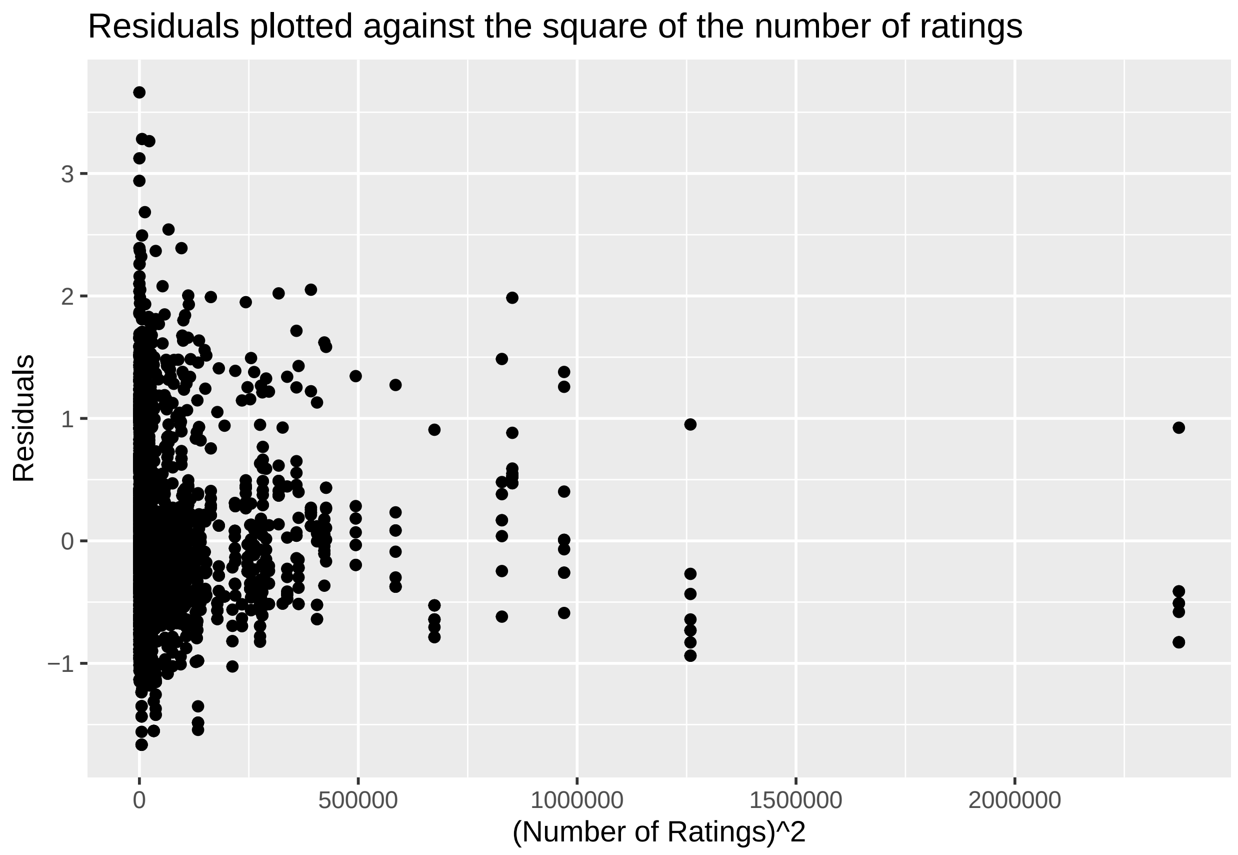
Conclusions: `ln(Price)^E = 5.0277153307 - 0.0000822596*(Number of Ratings)`. This is a log-level type regression, 5.0277153307 is the average 'ln(Price)' when 'Number of Ratings' is zero. The much more meaningful information is, however, that 'Price' is 0.00822596% lower on average for observations with one percent higher 'Number of Ratings'.

Exercise 3.

I'll examine the residuals from the first model (Price-Distance).

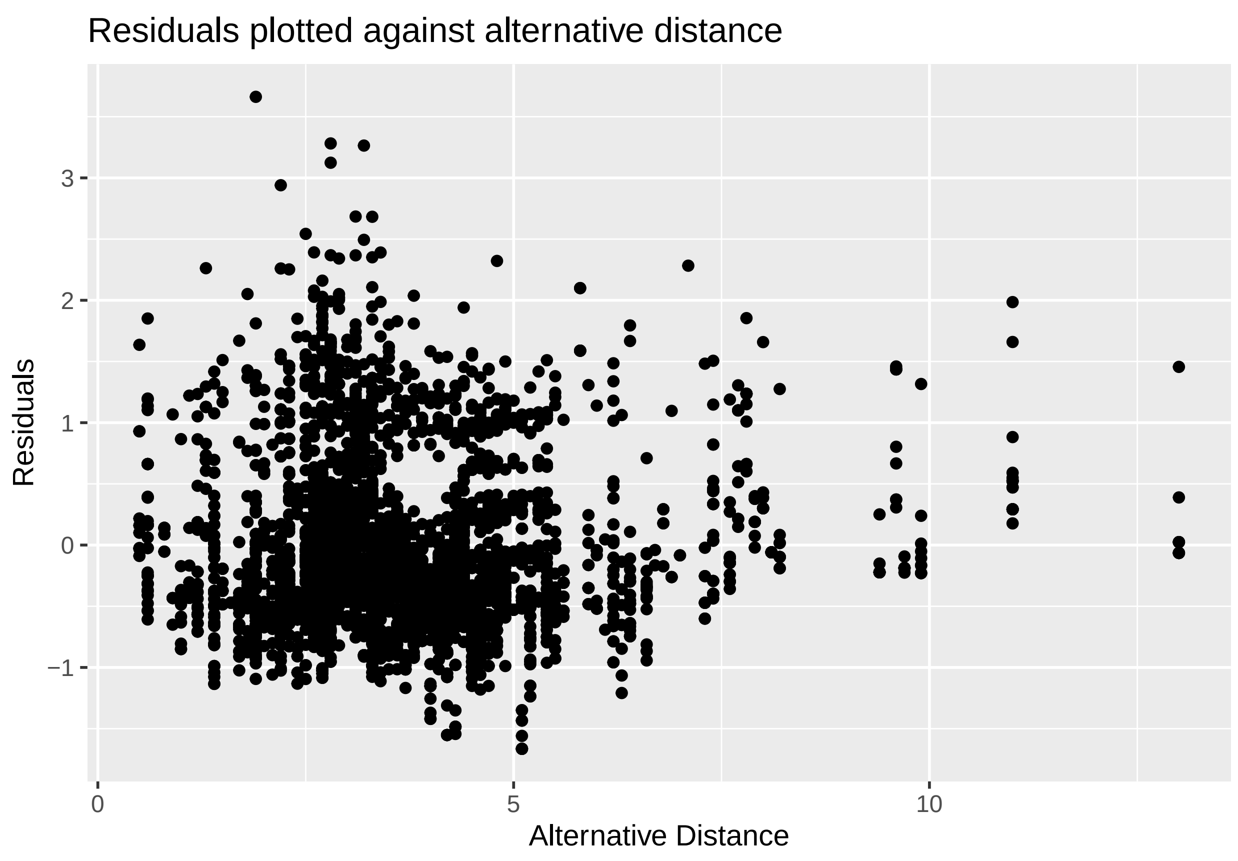
Residuals plotted against the square of the number of ratings.

```
ggplot(df, aes((vienna$rating_reviewcount)^2, df$resid)) +
  geom_point() +
  labs(x = "(Number of Ratings)^2",
       y = "Residuals",
       title = "Residuals plotted against the square of the number of ratings")
```



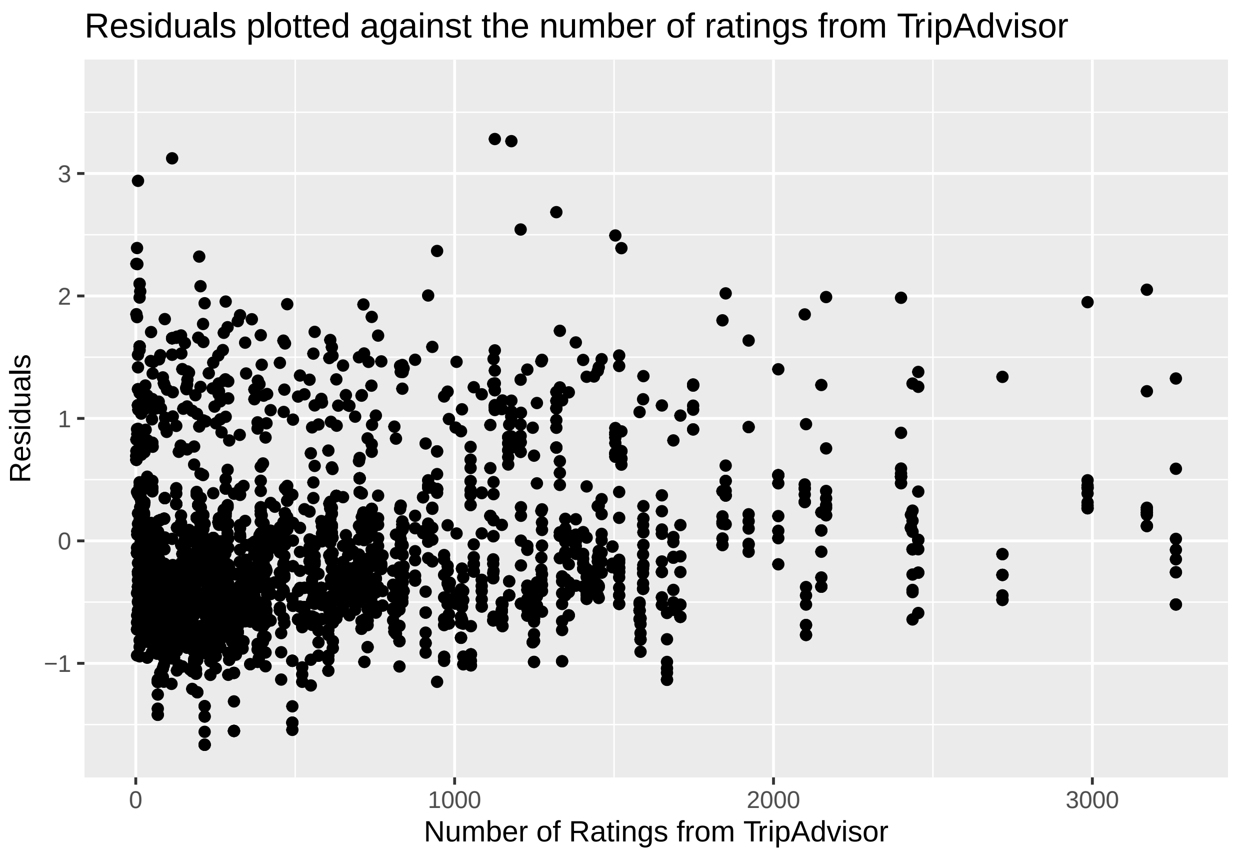
Residuals plotted against the so-called alternative distance which is the distance from ‘Center2’.

```r
ggplot(df, aes(vienna$distance_alter, df$resid)) +
  geom_point() +
  labs(x = "Alternative Distance",
       y = "Residuals",
       title = "Residuals plotted against alternative distance")
```



Residuals plotted against the number of ratings from TripAdvisor.

```
ggplot(df, aes(vienna$ratingta_count, df$resid)) +
  geom_point() +
  labs(x = "Number of Ratings from TripAdvisor",
       y = "Residuals",
       title = "Residuals plotted against the number of ratings from TripAdvisor")
```

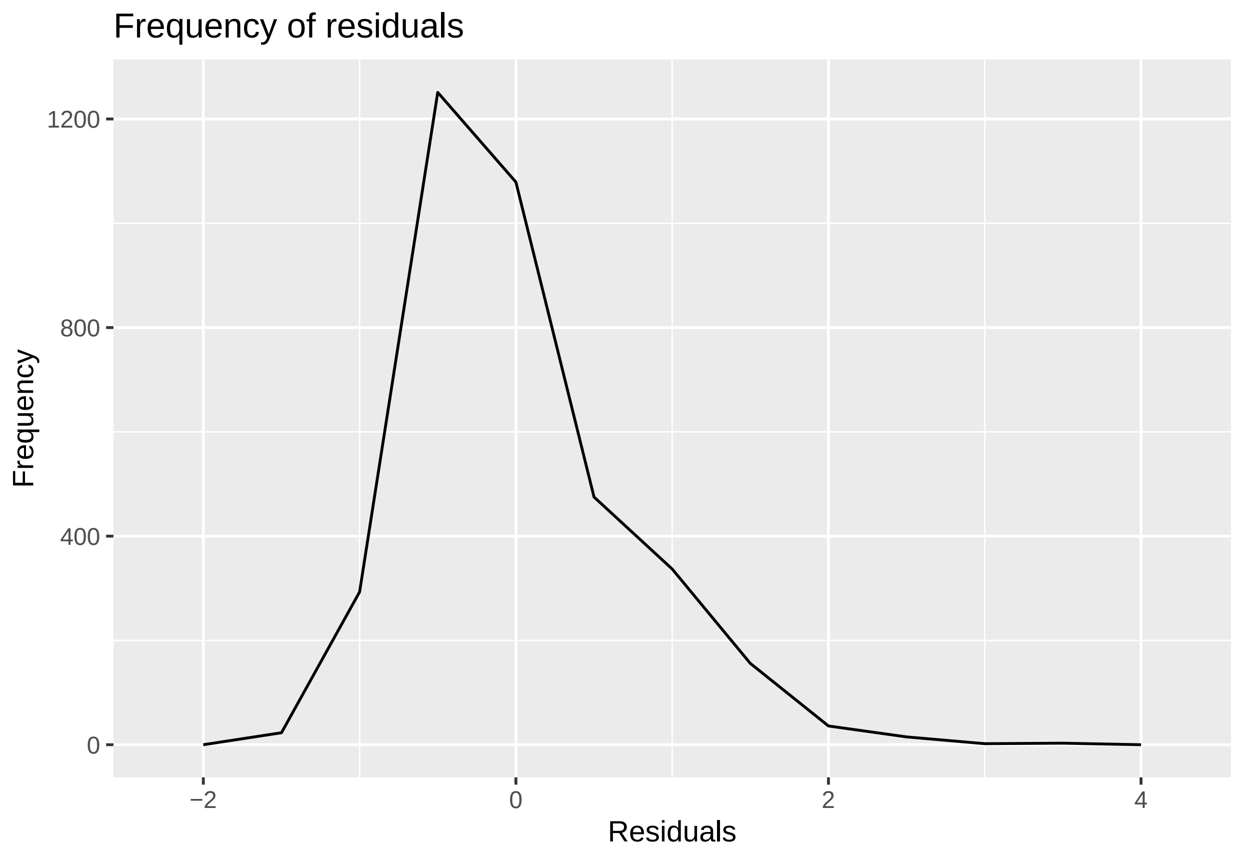


I reckon that at first glance in none of the graphs does the residual look random, but I'll examine it in 'Exercise 4'.

Exercise 4.

First, let's draw a frequency polygon that helps me to understand the spread of the residuals.

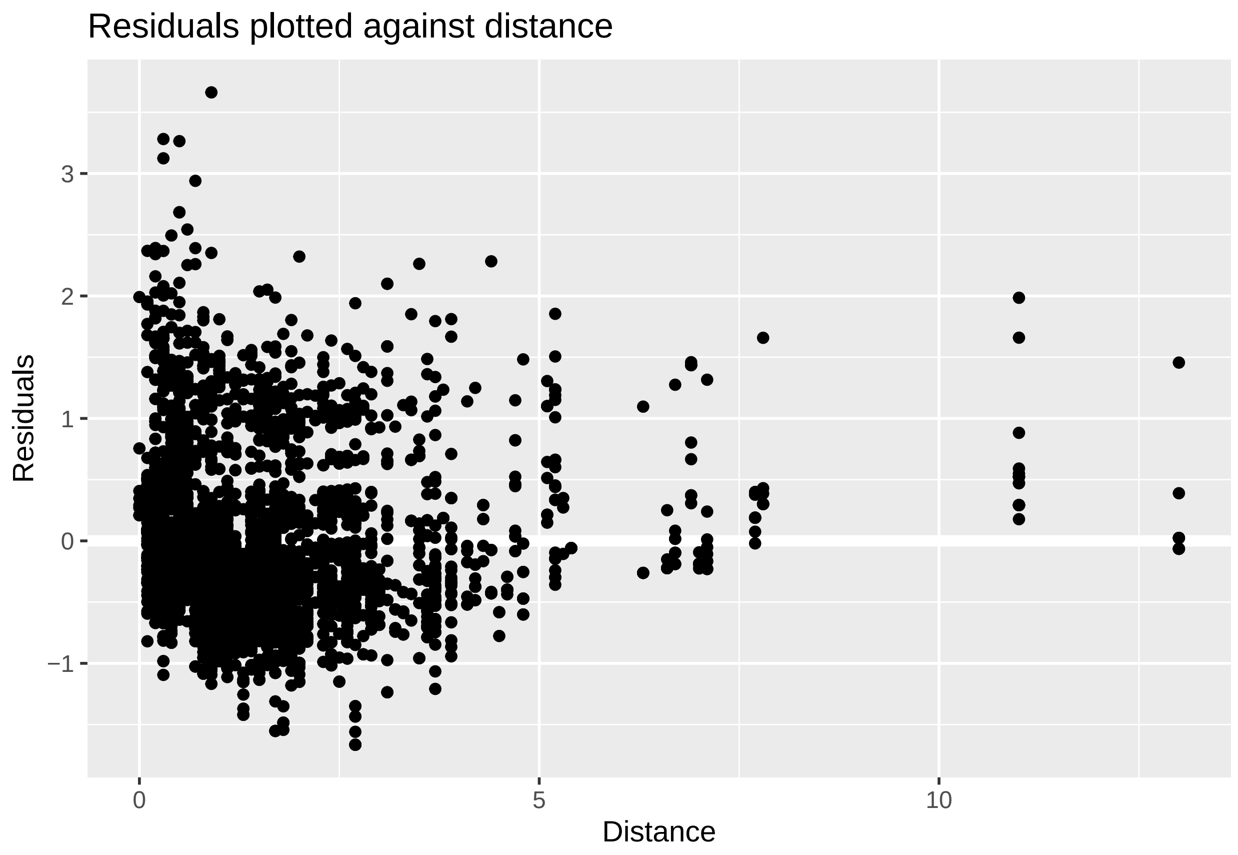
```r
ggplot(df, aes(df$resid)) +
  geom_freqpoly(binwidth = 0.5) +
  labs(x = "Residuals",
       y = "Frequency",
       title = "Frequency of residuals")
```



Based on the above graph we can be a bit more certain that residuals are not random as there should not be such a huge spike in the frequency of residual values, they should be more evenly distributed.

Second, let's create a plot using the residuals instead of the original predictor and plot it against 'Distance'.

```
ggplot(df, aes(df$distance, df$resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  labs(x = "Distance",
       y = "Residuals",
       title = "Residuals plotted against distance")
```



This doesn't look like random noise which suggests that my model has done a poor job of capturing the patterns in the dataset.

Exercise 5.

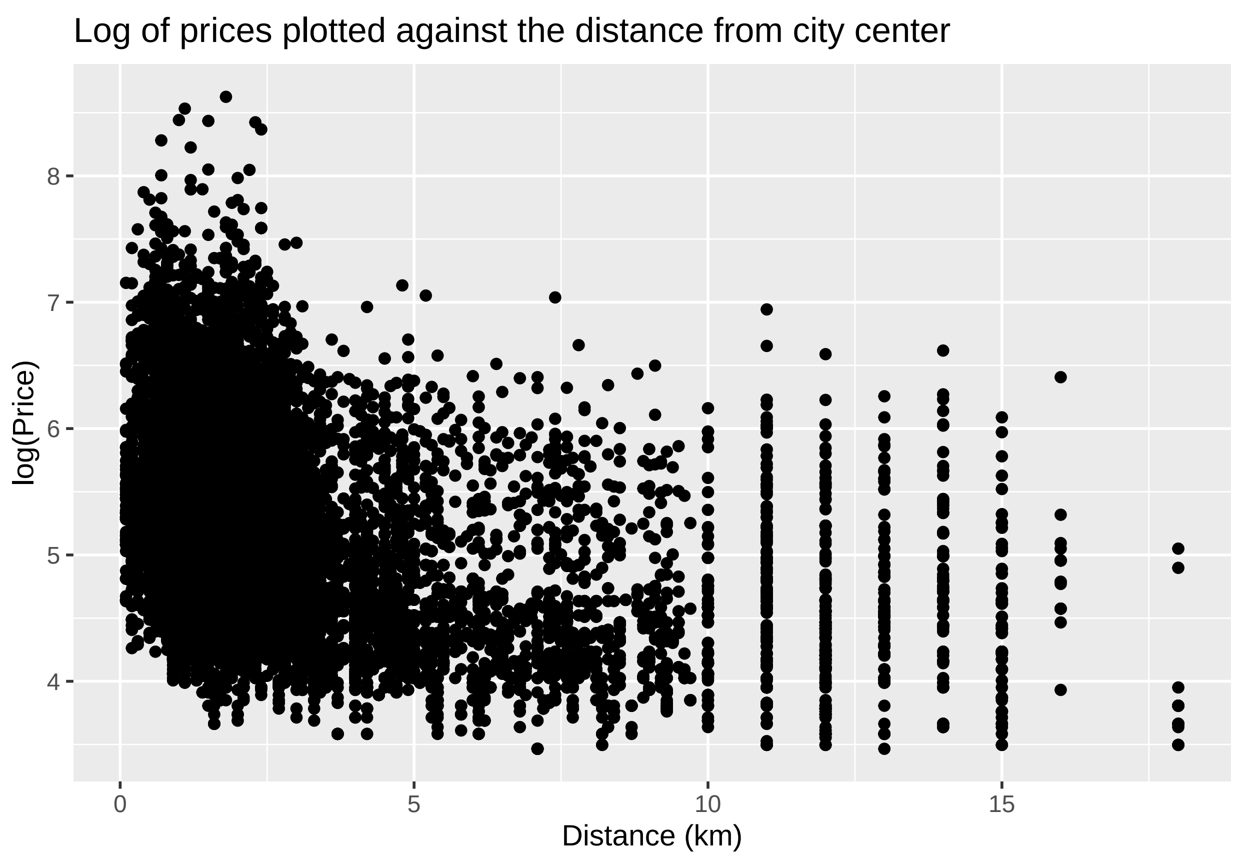
As none of the 3 models capture the patterns quite well, I chose the first model yet again.

Filter to Paris

```
paris <- hotels %>%
  filter(city == 'Paris')
```

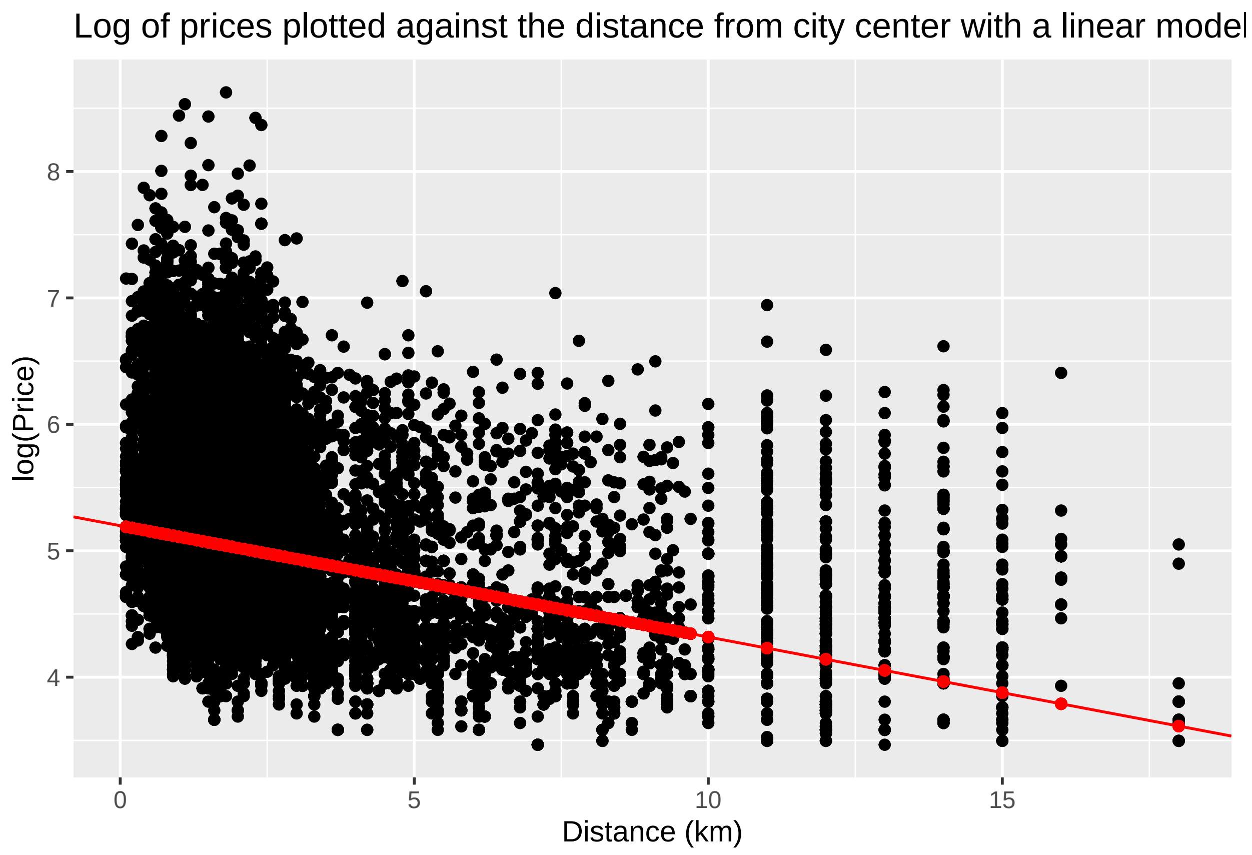
Plot the natural logarithm of prices against distance from the city center

```
paris_subset <- paris %>%
  transmute(price, distance)

paris_subset$price <- log(paris_subset$price)

paris_subset %>%
  na.exclude() %>%
  ggplot(mapping = aes(x = paris_subset$distance, y = paris_subset$price)) +
  geom_point() +
  labs(x = "Distance (km)",
       y = "log(Price)",
       title = "Log of prices plotted against the distance from city center")
```



Let’s apply ‘mod1’ from the Vienna models which aims to capture the pattern on the ‘Price’-‘Distance’ scatter plot.

```r
df_paris <- paris_subset %>%
  add_predictions(mod1) %>%
  add_residuals(mod1)

ggplot(df_paris, aes(df_paris$distance,df_paris$price)) +
  geom_point() +
  geom_point(aes(df_paris$distance,df_paris$pred), color = "red") +
  geom_abline(aes(intercept = coef(mod1)[1], slope = coef(mod1)[2]), color = "red") +
  labs(x = "Distance (km)",
       y = "log(Price)",
       title = "Log of prices plotted against the distance from city center with a linear model")
```



In order to see how well the model does, let's check the R-squared.

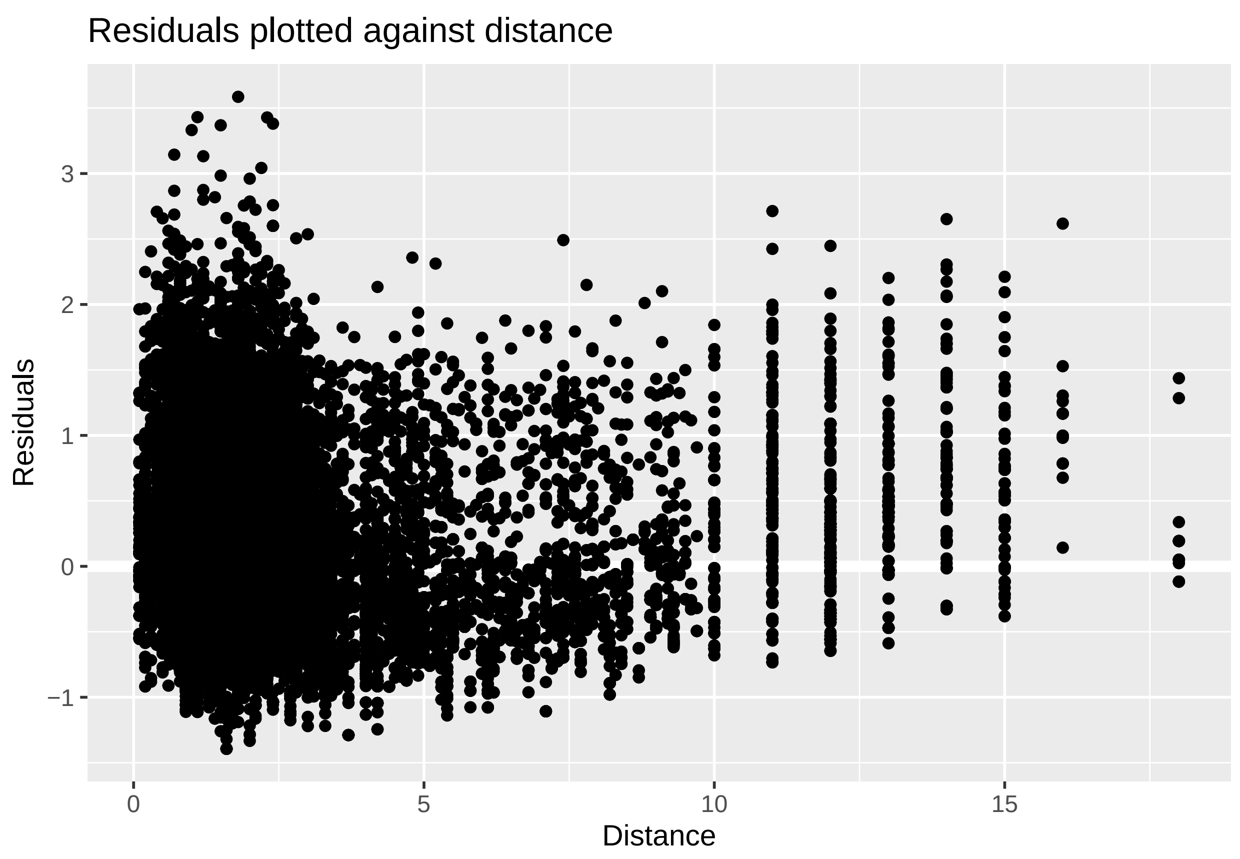
```r
summary(mod1)$r.squared
```

```
## [1] 0.04082566
```

As we can see 4.082566% of the variation in 'Price' is captured by the regression, and rest is left for residual variation. Therefore, there is plenty of room for improvement in terms of the model.

If we check the residuals and plot them against 'Distance', once again what we see is unlike random noise which suggests that the model has done a poor job of capturing the patterns in the dataset.

```r
ggplot(df_paris, aes(df_paris$distance, df_paris$resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  labs(x = "Distance",
       y = "Residuals",
       title = "Residuals plotted against distance")
```



Exercise 6.

I will reestimate 'mod1' using data on hotels in Barcelona.

Filter to Barcelona

```
barcelona <- hotels %>%
  filter(city == 'Barcelona')
```

Picking subset

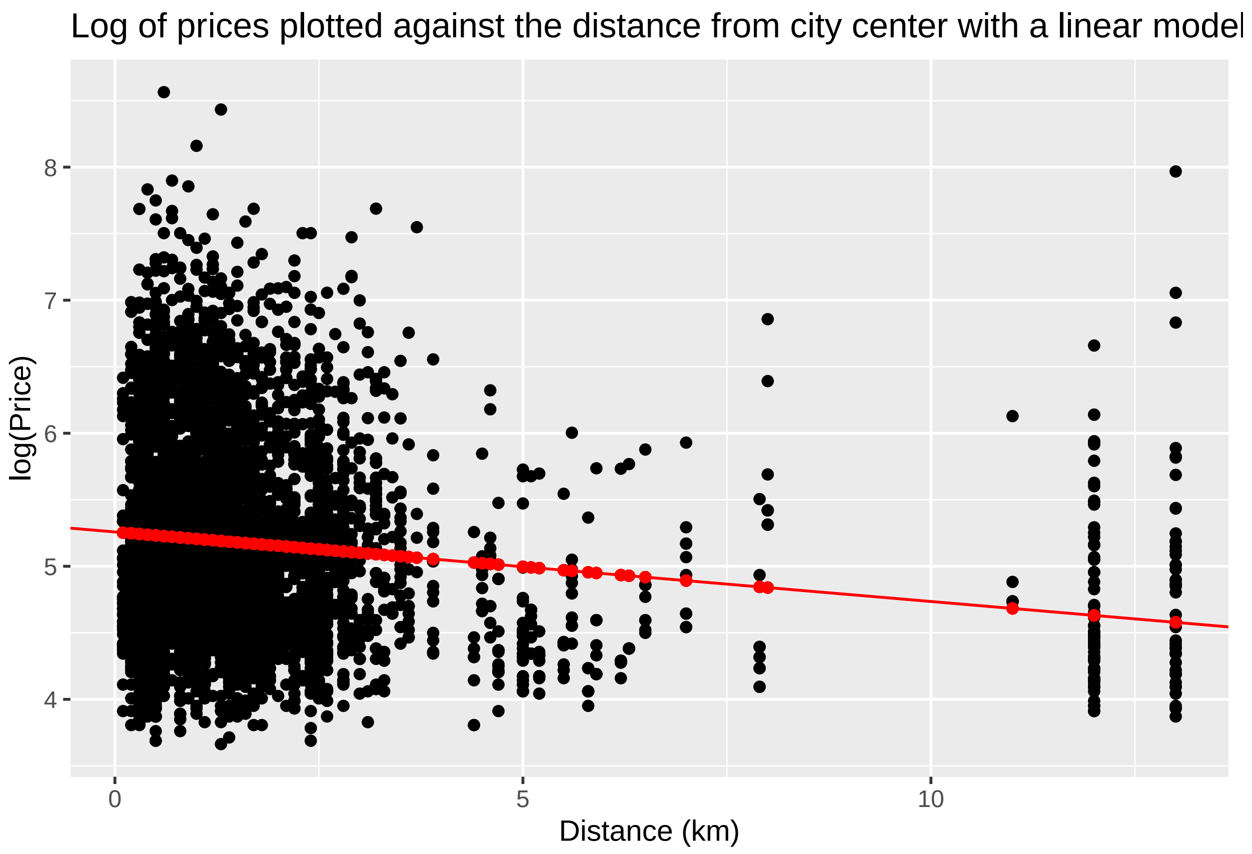
```
barcelona_subset <- barcelona %>%
  transmute(price, distance)

barcelona_subset$price <- log(barcelona_subset$price)
```

Model reestimation

```
mod1_re <- lm(price ~ distance, data = barcelona_subset)

coef(mod1_re)
```

```
## (Intercept)    distance
##  5.25818008 -0.05226644
```

```r
df_bar <- barcelona_subset %>%
  add_predictions(mod1_re) %>%
  add_residuals(mod1_re)

ggplot(df_bar, aes(df_bar$distance,df_bar$price)) +
  geom_point() +
  geom_point(aes(df_bar$distance,df_bar$pred), color = "red") +
  geom_abline(aes(intercept = coef(mod1_re)[1], slope = coef(mod1_re)[2]), color = "red") +
  labs(x = "Distance (km)",
       y = "log(Price)",
       title = "Log of prices plotted against the distance from city center with a linear model")
```



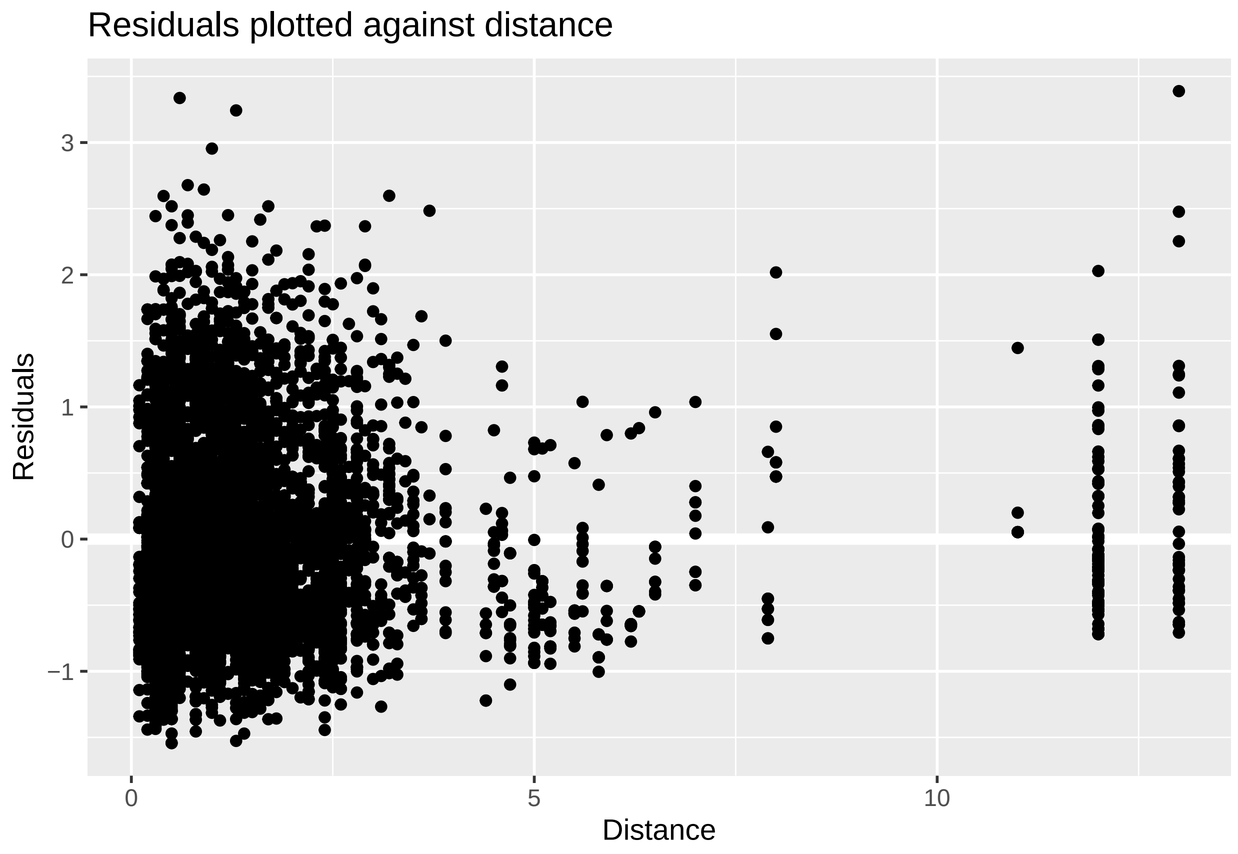
Check the change of the parameter values:

- Intercept: For Vienna it was 5.19812681, now in case of Barcelona it is 5.25818008 which is approx. 1.16% higher than the previous one
- Slope: For Vienna it was -0.08804268, now in case of Barcelona it is -0.05226644 which is approx. 40.6% lower than the previous one

To determine how well the model does now, let's check the residuals

For this I created a plot using the residuals instead of the original predictor and plotted it against 'Distance'.

```r
ggplot(df_bar, aes(df_bar$distance, df_bar$resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  labs(x = "Distance",
       y = "Residuals",
       title = "Residuals plotted against distance")
```



This doesn't look like random noise which suggests that the model has yet again done a poor job of capturing the patterns in the dataset. But is it worse than the previous one? To determine this, let's check the R-squared.

```
summary(mod1)$r.squared # Vienna
```

```
## [1] 0.04082566
```

```
summary(mod1_re)$r.squared # Barcelona
```

```
## [1] 0.02333772
```

As we have already seen, 4.082566% of the variation in 'Price' is captured by the Vienna-model and now it can be seen that only 2.333772% of the variation in 'Price' is captured by the Barcelona-model which means that the reestimated model is actually worse than the previous one.

If I had to summarise the difference between the two cities I would choose the slope coefficients as they are quite meaningful. For Vienna 'Price' is 8.804268% lower on average for observations having one unit higher 'Distance' whereas for Barcelona 'Price' is 5.226644% lower on average for observations having one unit higher 'Distance.