Assignment 9

Kornel Kovacs

11/09/2019

I created a project and placed the csv files in its root from where I read them in.

Exercise 1:

I realize that it is the `hotel_id` column on which I should join the two tables. Notice, that `df_hotel` will become panel data as it contains more observations across time for each hotel.

```
df_hotel<- df_price %>%
  left_join(df_features, by = "hotel_id")
```

These 3 rows do not have feature data in `df_features`.

```
df_price %>%
  anti_join(df_features, by = "hotel_id")
```

```
## # A tibble: 3 x 10
##   hotel_id price offer offer_cat  year month weekend holiday nnights
##      <dbl> <dbl> <dbl> <chr>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>
## 1        2   119     0 0% no of~  2017    11       0       0       1
## 2        2   119     0 0% no of~  2017    12       0       1       1
## 3        2   547     0 0% no of~  2017    12       0       1       4
## # ... with 1 more variable: scarce_room <dbl>
```

Exercise 2

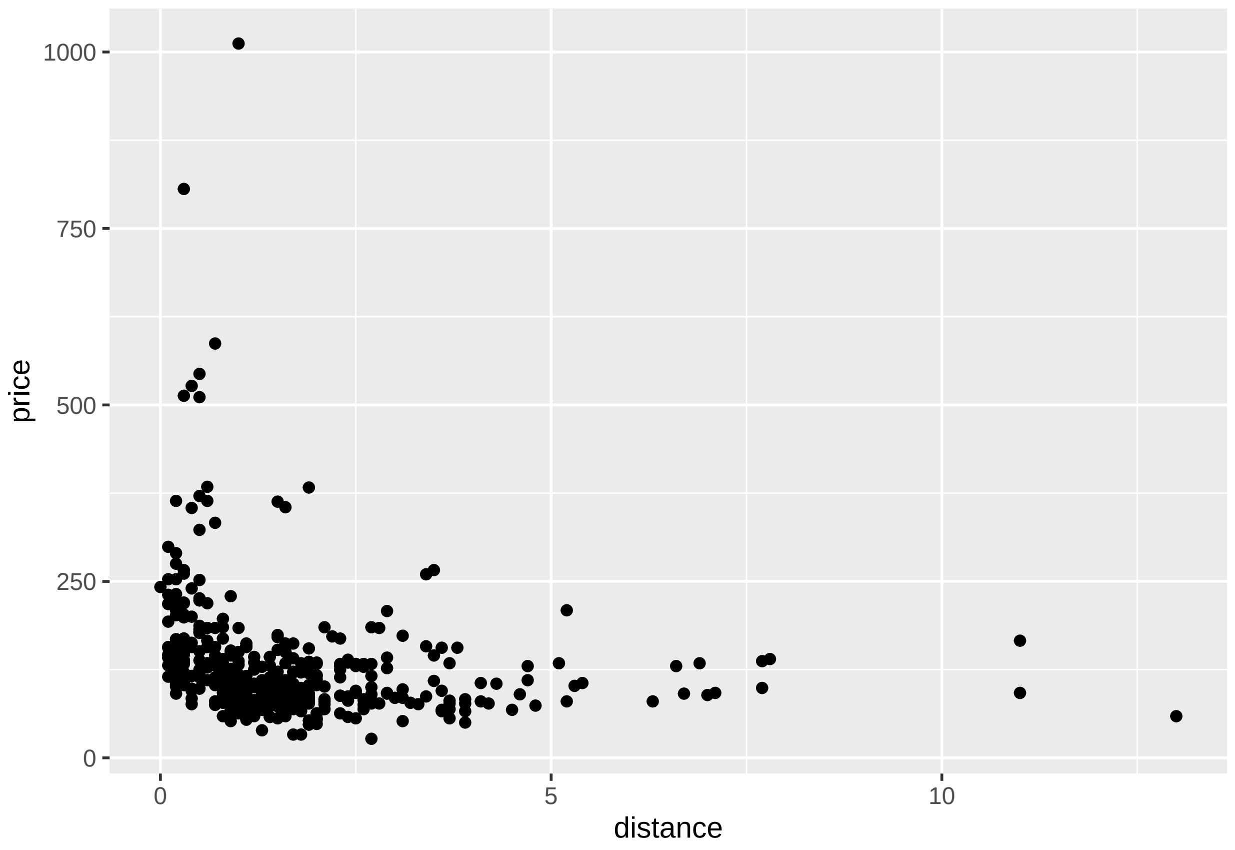
First, I read the vienna dataset in.

Plotting price against rationally relevant explanatory variables and estimating linear models.

I print out the coefficient, the intercept and the R squared values as well as the plots.

Price and distance

```
p <- ggplot(data = df_vienna,
            mapping = aes(x = distance, y = price))
p + geom_point()
```



```r
price_dist <- lm(price ~ distance, data = df_vienna)
summary(price_dist)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 151.29236   6.255225 24.186557 5.731224e-82
## distance    -12.01145   2.719123 -4.417399 1.268452e-05
```

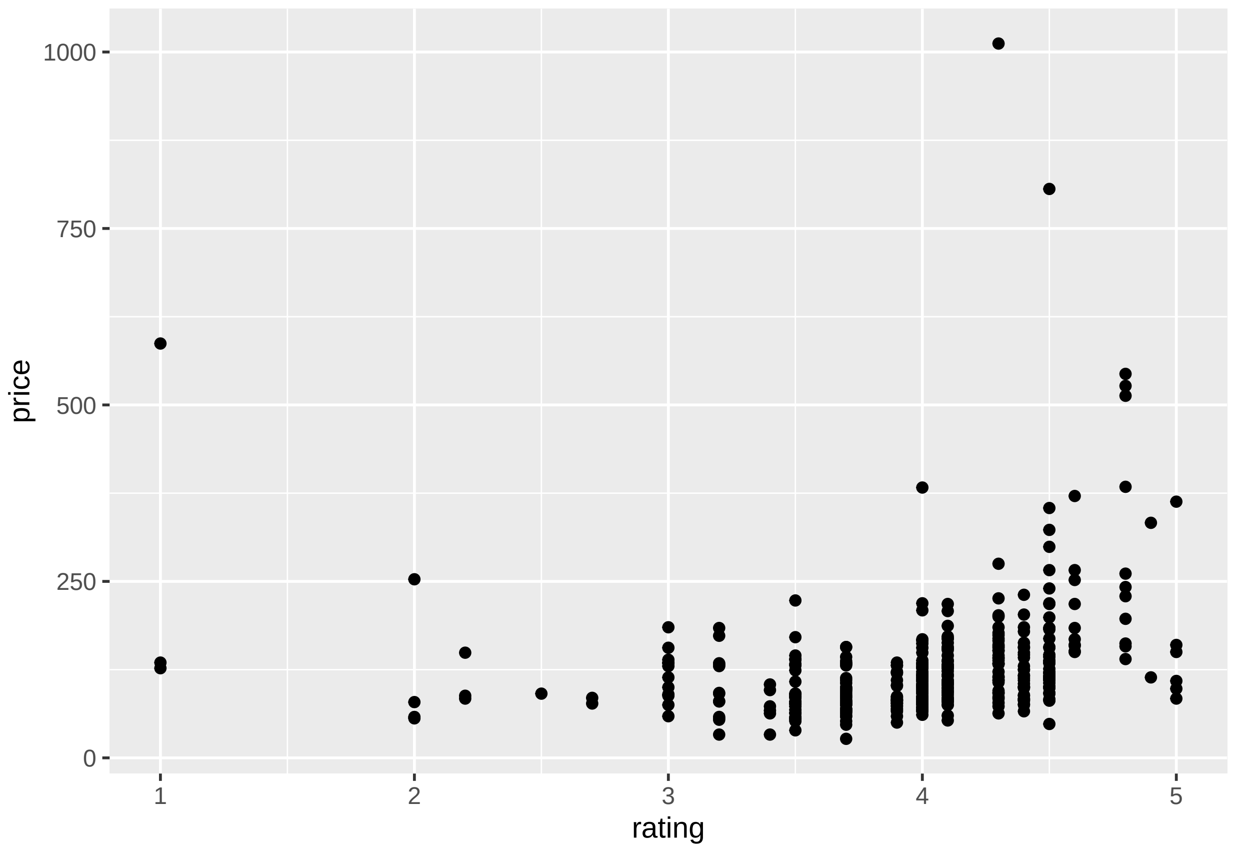
```r
print(paste0("R squared: ", summary(price_dist)$r.squared))
```

```
## [1] "R squared: 0.0437998376417376"
```

Price and ratings

```r
p <- ggplot(data = df_vienna,
            mapping = aes(x = rating, y = price))
p + geom_point()
```

```
## Warning: Removed 35 rows containing missing values (geom_point).
```



```r
price_ratings <- lm(price ~ rating, data = df_vienna)
summary(price_ratings)$coefficients
```

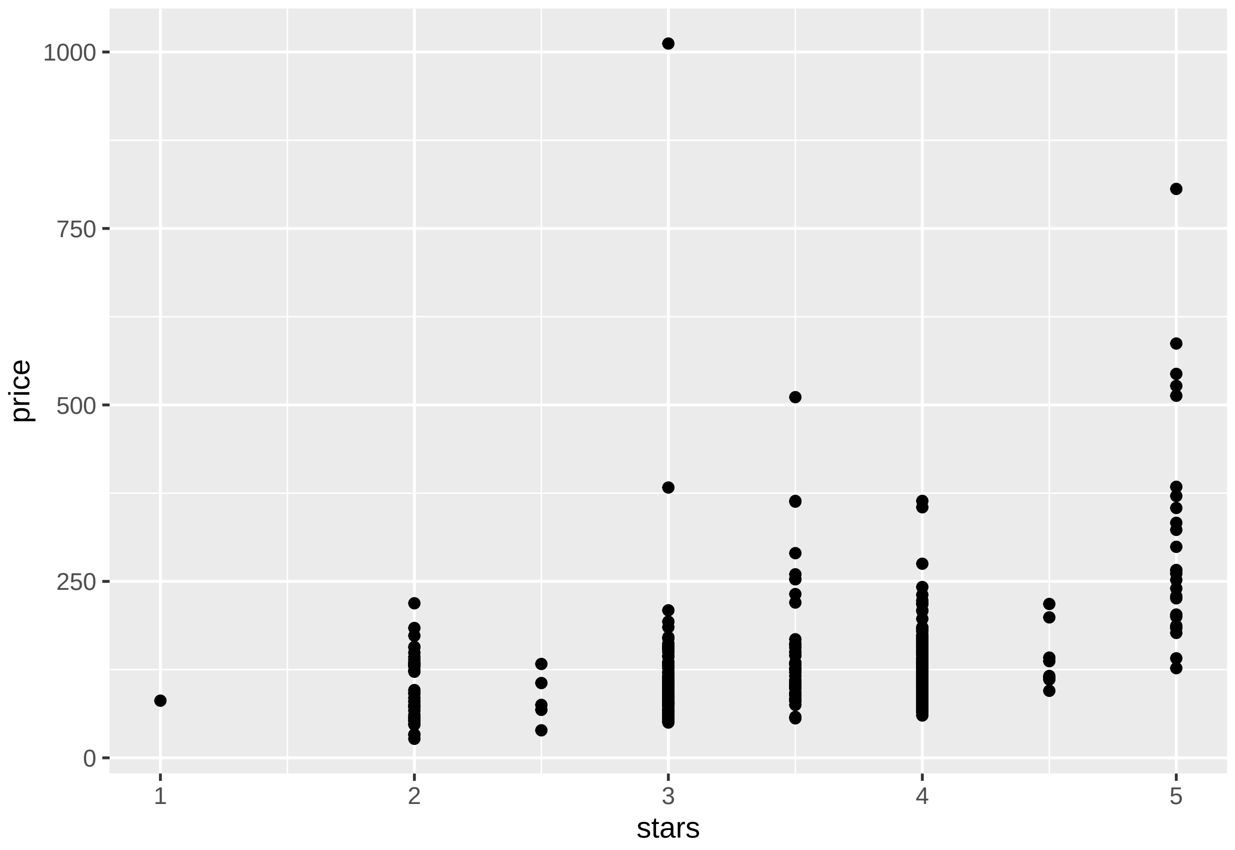
```
##              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  1.923557  30.949866 0.06215075 9.504745e-01
## rating      31.663490   7.713071 4.10517309 4.920836e-05
```

```r
print(paste0("R squared: ", summary(price_ratings)$r.squared))
```

```
## [1] "R squared: 0.0413199583539873"
```

Price and stars

```r
p <- ggplot(data = df_vienna,
            mapping = aes(x = stars, y = price))
p + geom_point()
```



```
price_stars <- lm(price ~ stars, data = df_vienna)
summary(price_stars)$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -17.27814  18.829754 -0.9175977 3.593487e-01
## stars        43.27894   5.349156  8.0907984 6.243588e-15
```

```
print(paste0("R squared: ", summary(price_stars)$r.squared))
```

```
## [1] "R squared: 0.133196767062536"
```

It is the variable `stars` that seems to have the most explanatory power, however, when looking at the graph, we can observe that the relationship does not seem to be linear.

Exercise 3

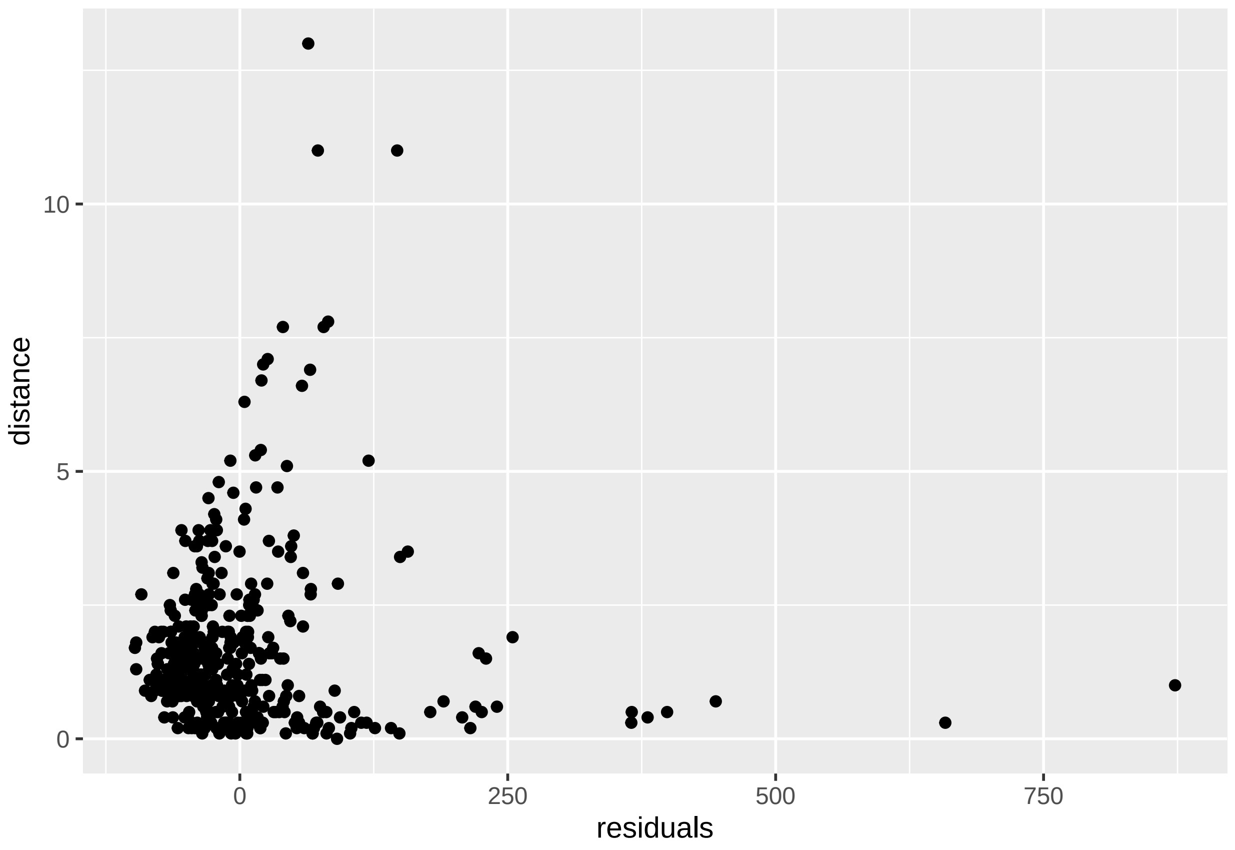
I will go with `price_dist` model in this exercise.

First, I extract the residuals from my favourite model.

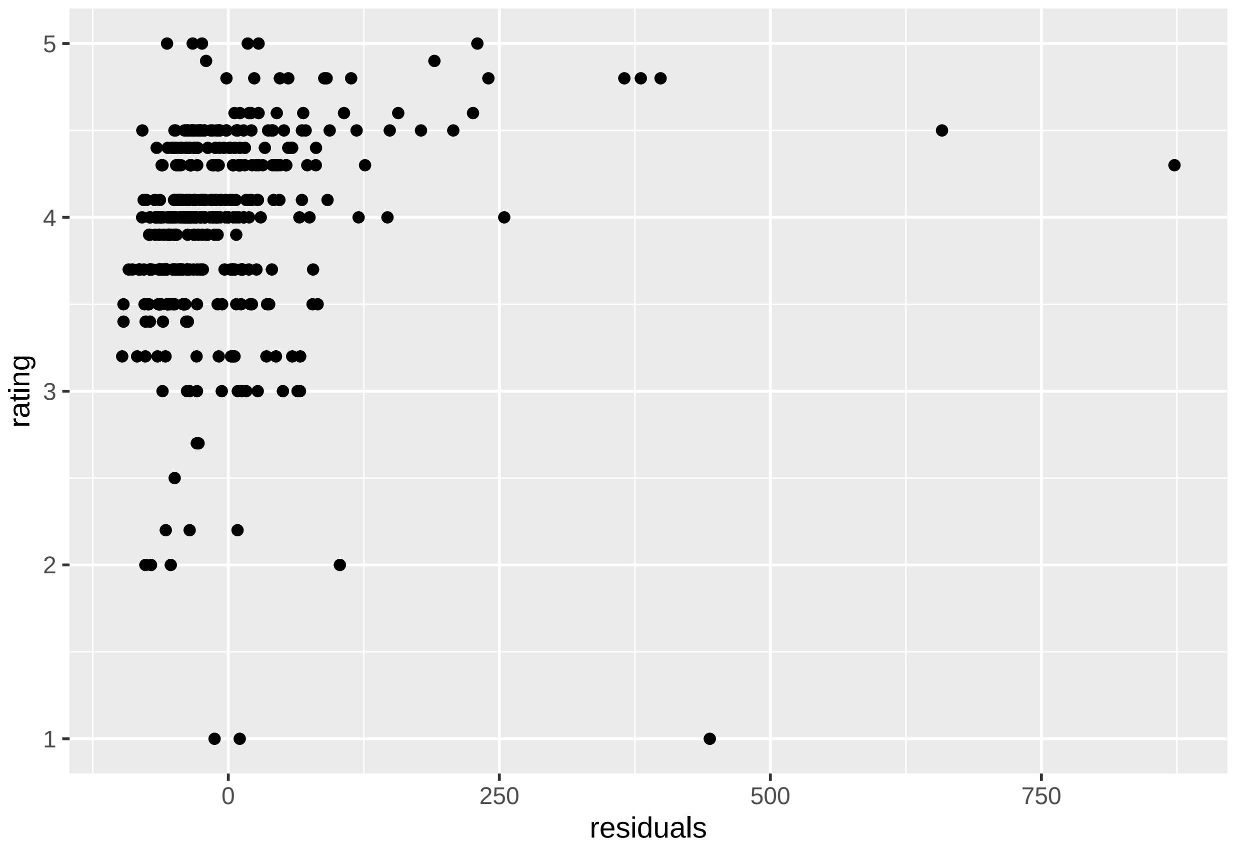
```
df_vienna$res <- resid(price_dist)
```

Then, I plot it against some vairables.

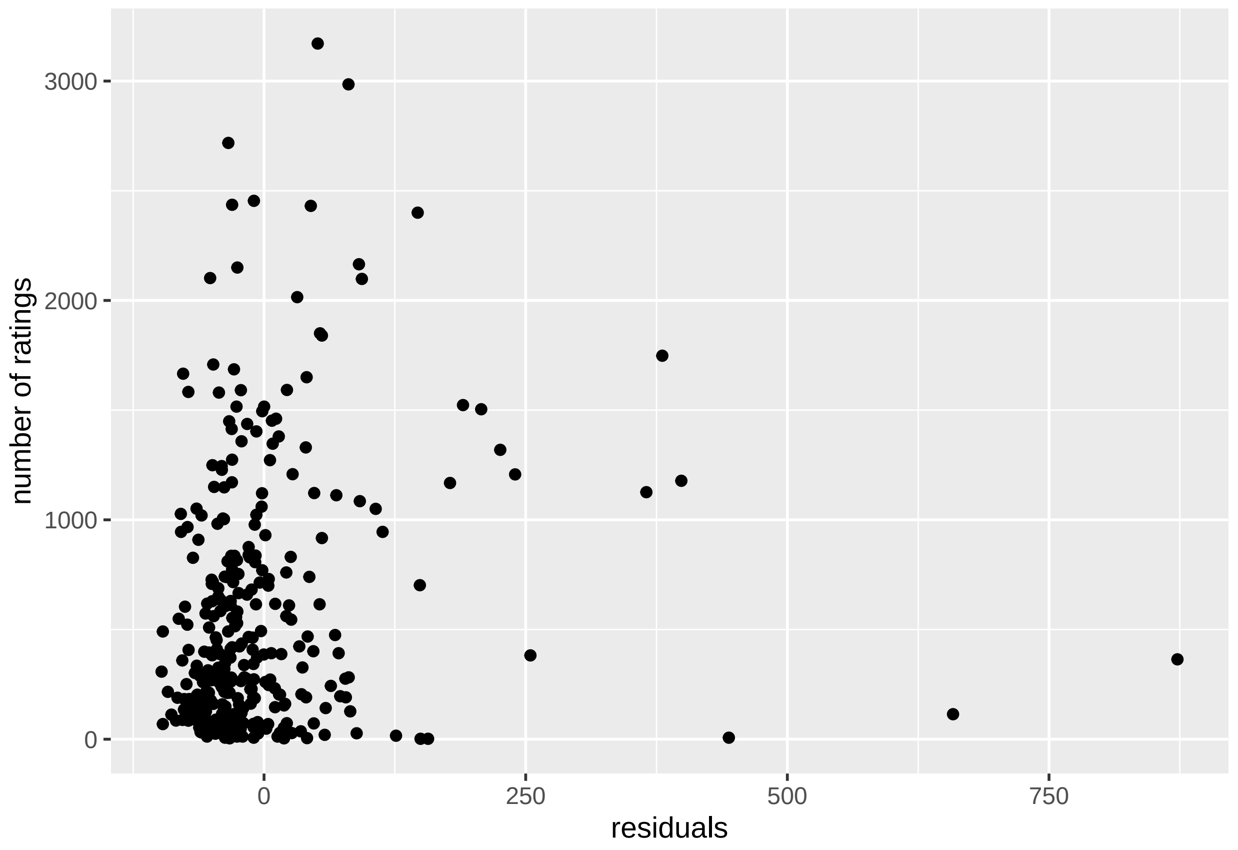
```
p <- ggplot(data = df_vienna,
            mapping = aes(x = res, y = distance))
p + geom_point() + xlab("residuals")
```



```
p <- ggplot(data = df_vienna,
            mapping = aes(x = res, y = rating))
p + geom_point() + xlab("residuals")
```



```
p <- ggplot(data = df_vienna,
            mapping = aes(x = res, y = ratingta_count))
p + geom_point() + xlab("residuals") + ylab("number of ratings")
```



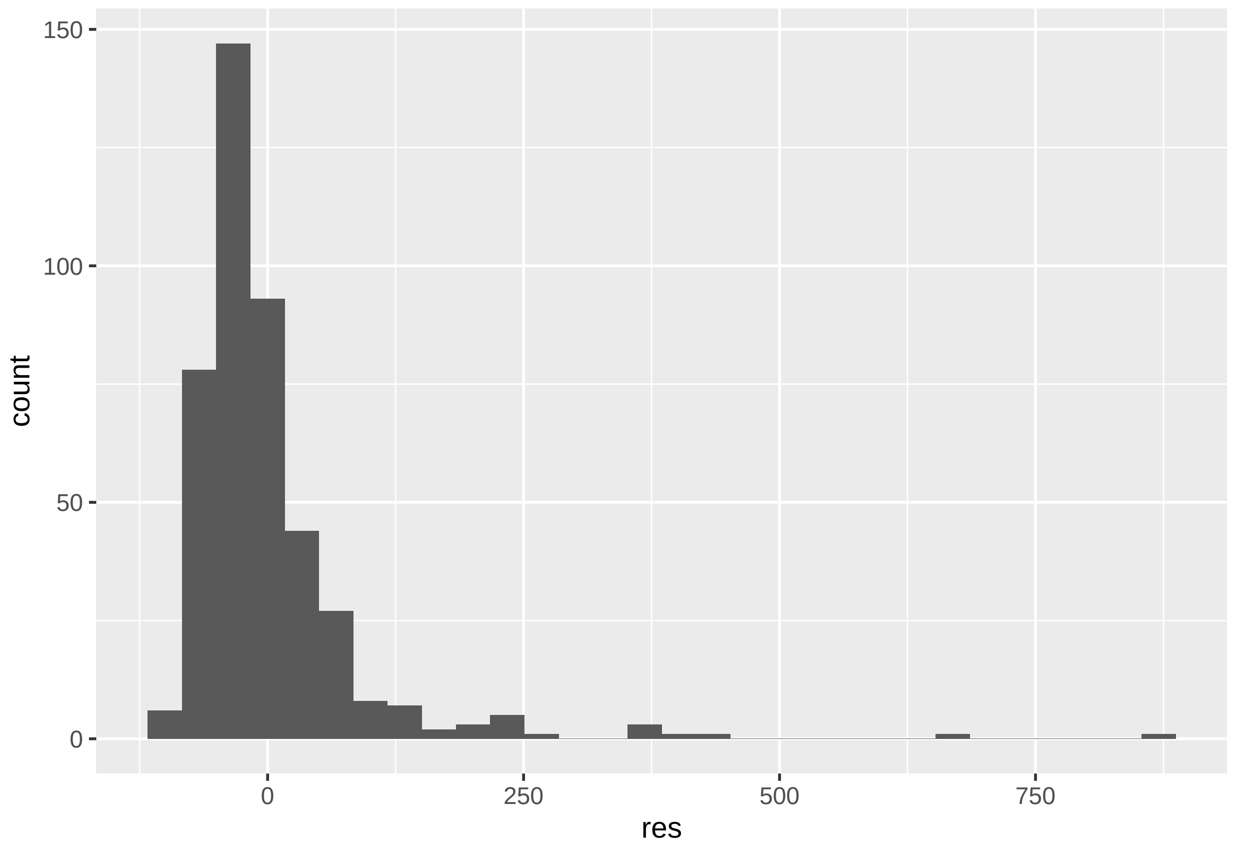
In every single case, there seems to be a tendency. Residuals do not look to be randomly distributed when plotting against the above variables.

Exercise 4

We want to have a sense of the distribution of our residuals. When estimating linear regressions, a key is that we have **homoskedasticity**.

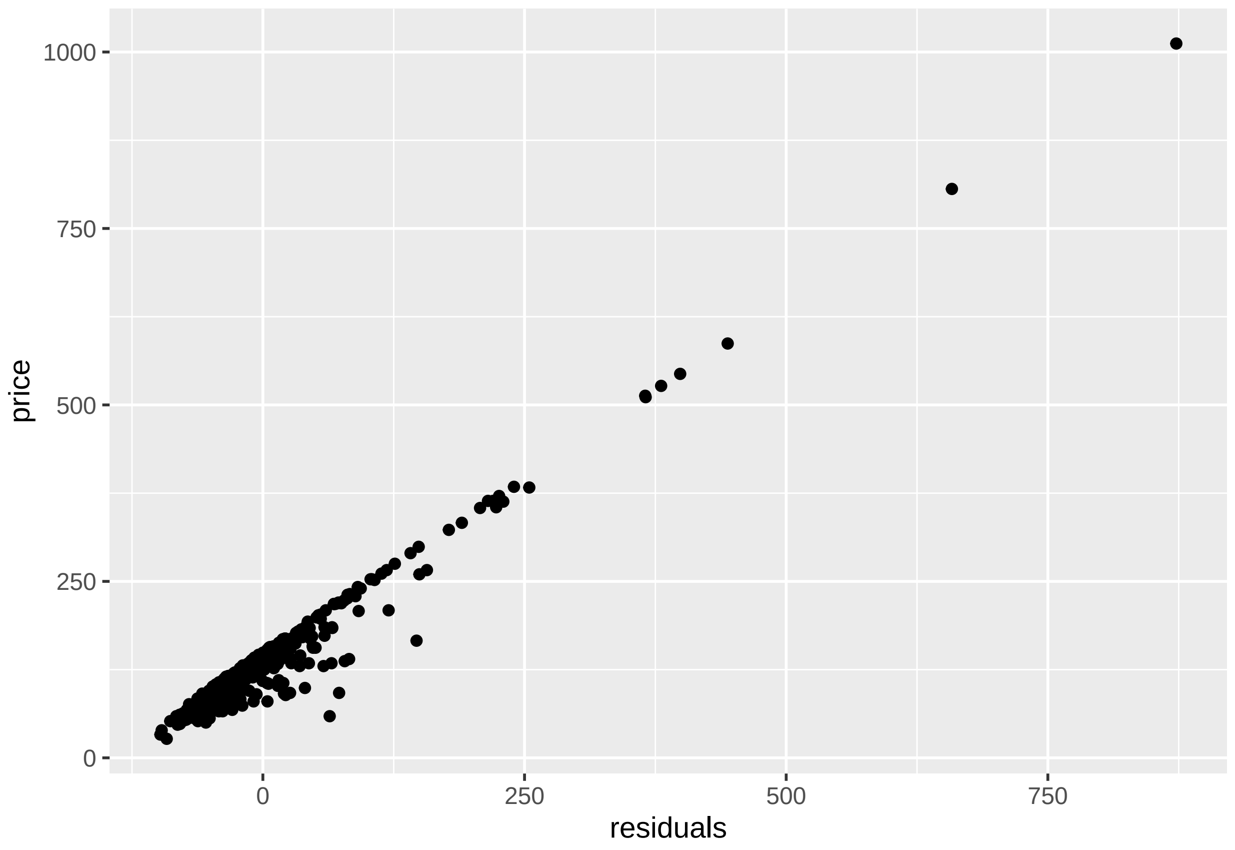
First, I plot a histogram to have a better understanding of the distribution of residuals. It has some extreme values and a long right tail, but most of the values are around 0.

```r
ggplot(df_vienna, aes(res)) +
    geom_histogram()
```



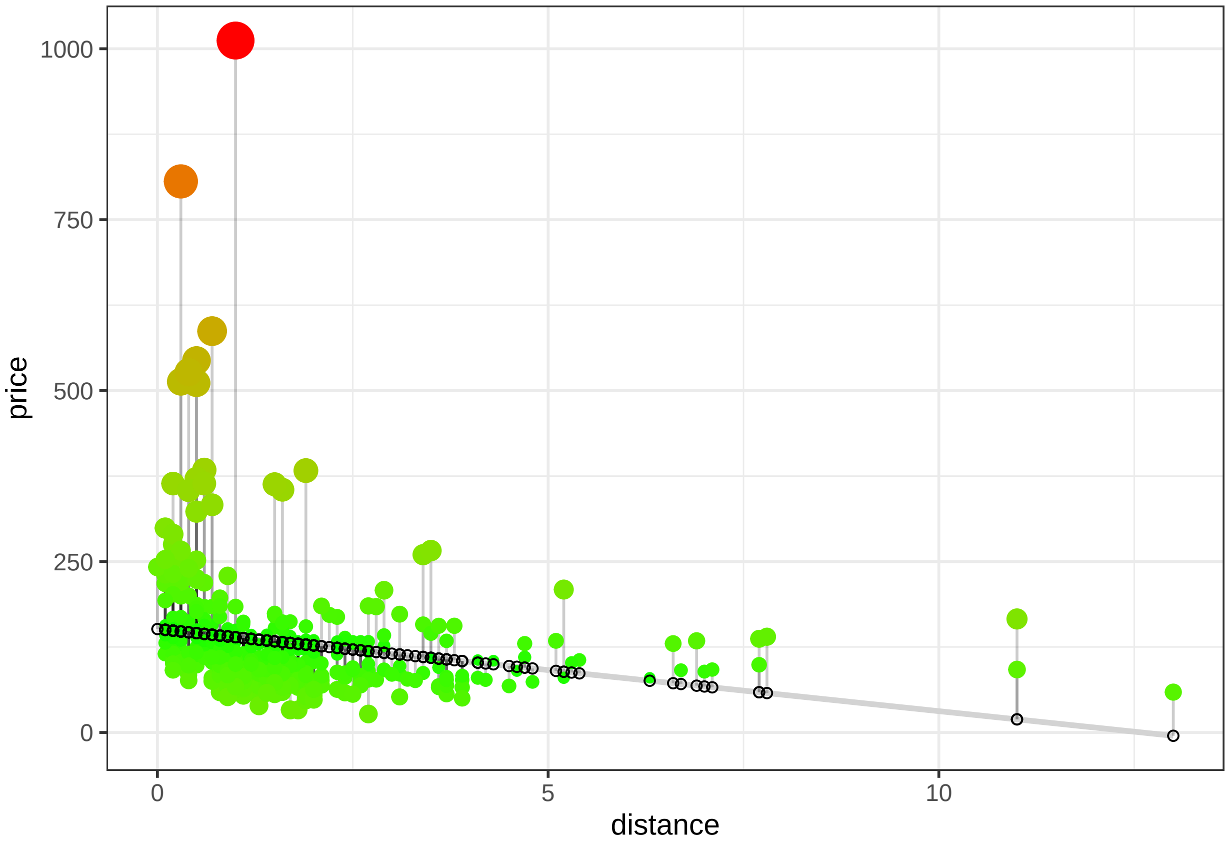
When we plot the residuals against the y variable we should see a linear(ish) relationship, which we quite do.

```
p <- ggplot(data = df_vienna,
            mapping = aes(x = res, y = price))
p + geom_point() + xlab("residuals")
```



I can also plot a graph reflecting to the greatness of differences between predicted values and actual values. Kudos to Rpubs

```
df_vienna$predicted <- predict(price_dist)
ggplot(df_vienna, aes(x = distance, y = price)) +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey") +
  geom_segment(aes(xend = distance, yend = predicted), alpha = .2) +
  geom_point(aes(color = abs(res), size = abs(res))) +
  scale_color_continuous(low = "green", high = "red") +
  guides(color = FALSE, size = FALSE) +
  geom_point(aes(y = predicted), shape = 1) +
  theme_bw()
```



There are more sophisticated ways to deal with heteroskedasticity. I rather went with showing some plots.

Exercise 5

First, I get rid of its panel data nature by filtering to a specific year and month which I chose arbitrarily. As for the city, I chose Amsterdam.

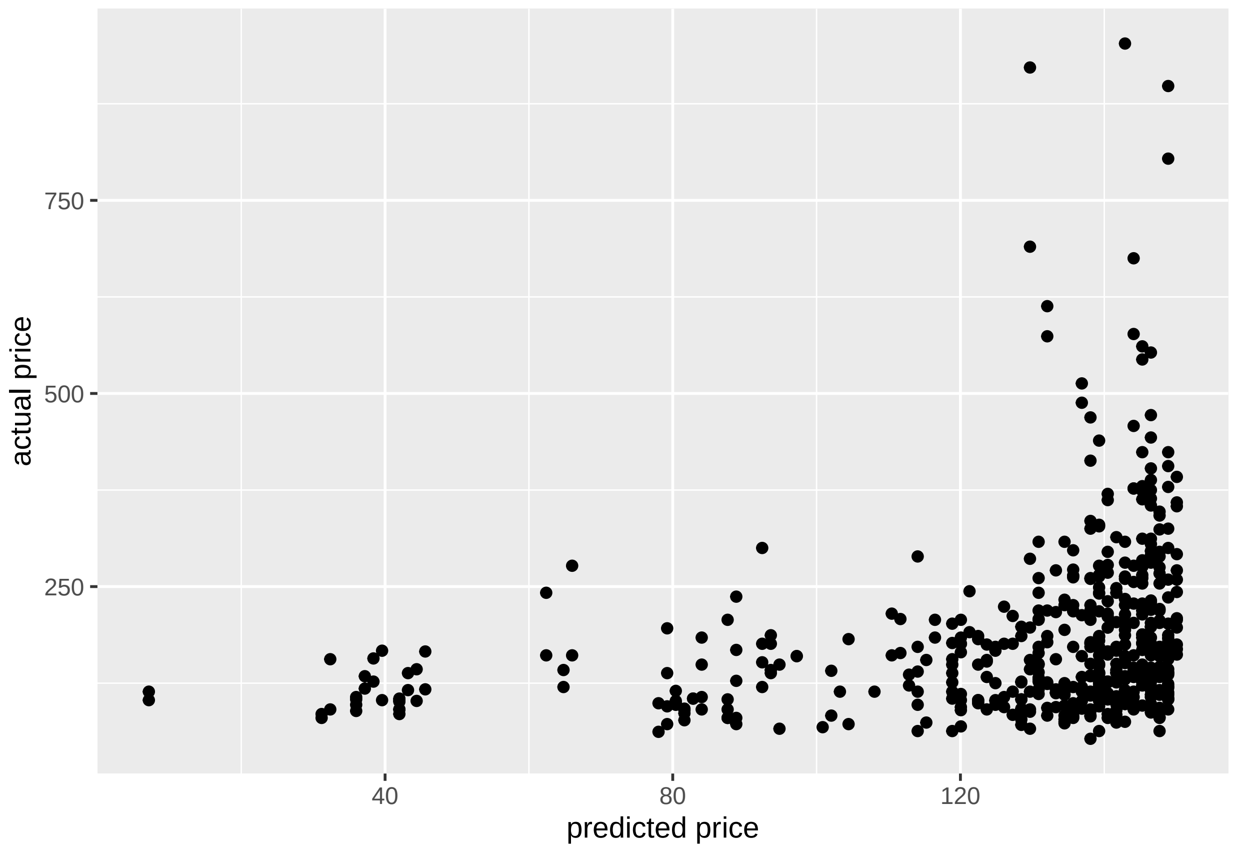
```
df_ams <- df_hotel %>%
  filter(city == 'Amsterdam') %>%
  filter(year == 2017) %>%
  filter(month == 11)
```

First, I predict the values estimated from the Vienna model of price and distance using AMS data.

```
df_ams$predicted <- predict(price_dist, df_ams)
```

A scatterplot of estimated and actual values shows that these two cities have different characteristics. Most importantly, AMS is more expensive.

```
p <- ggplot(data = df_ams,
            mapping = aes(x = predicted, y = price))
p + geom_point() + xlab("predicted price") + ylab("actual price")
```

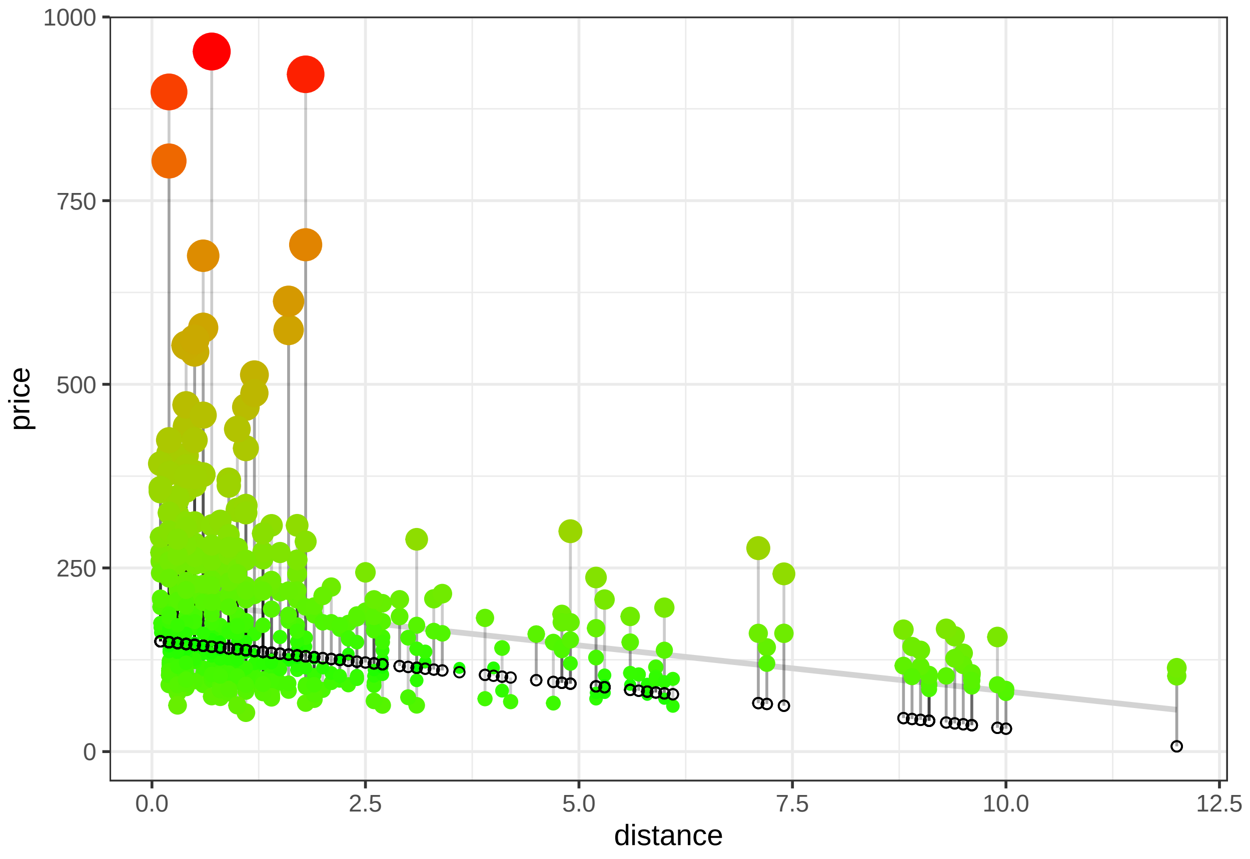


Then, we calculate the residual values for AMS.

```
df_ams$resid_ams <- df_ams$price - df_ams$predicted
```

As we have the residuals, we can plot a nice graph well illustrating the differences between predicted values and actual ones. The grey line illustrates where the regression line should be if we used AMS data for estimation. Again, kudos to Rpubs

```
ggplot(df_ams, aes(x = distance, y = price)) +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey") +
  geom_segment(aes(xend = distance, yend = predicted), alpha = .2) +
  geom_point(aes(color = abs(resid_ams), size = abs(resid_ams))) +
  scale_color_continuous(low = "green", high = "red") +
  guides(color = FALSE, size = FALSE) +
  geom_point(aes(y = predicted), shape = 1) +
  theme_bw()
```



We can conclude that the Vienna model is underestimating prices in AMS.

Exercise 6.

I reestimate my favourite model with AMS data.

```
price_dist_ams <- lm(price ~ distance, data = df_ams)
summary(price_dist_ams)$coefficients
```

```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 207.1471   6.356928 32.586037 7.379909e-129
## distance    -12.5110   2.076465 -6.025144  3.156916e-09
```

```
print(paste0("R squared: ", summary(price_dist_ams)$r.squared))
```

```
## [1] "R squared: 0.0638786080736448"
```

I a reminder, I have the Vienna model here:

```
price_dist <- lm(price ~ distance, data = df_vienna)
summary(price_dist)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 151.29236   6.255225 24.186557 5.731224e-82
## distance    -12.01145   2.719123 -4.417399 1.268452e-05
```

```
print(paste0("R squared: ", summary(price_dist)$r.squared))
```

```
## [1] "R squared: 0.0437998376417376"
```

We have a higher R squared value, but it doesn't mean much in terms of comparison, because the y variable is actually different. The intercept is greater, which means that prices right at the center of the city are greater in AMS. However, they tend to be lower more when you are a further kilometer away from there.