midterm-projv2

Yudong Wang

4/18/2021

Extra Analysis

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(geoR)
```

```
## Warning: package 'geoR' was built under R version 4.0.5
```

```
## ----------------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.8-1 (built on 2020-02-08) is now loaded
## ----------------------------------------------------------------
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.5
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
# read data
mydata = read_csv("Airbnb_NYC_2019.csv")
```

```
##
```

```
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   name = col_character(),
##   host_id = col_double(),
##   host_name = col_character(),
##   neighbourhood_group = col_character(),
##   neighbourhood = col_character(),
##   latitude = col_double(),
##   longitude = col_double(),
##   room_type = col_character(),
##   price = col_double(),
##   minimum_nights = col_double(),
##   number_of_reviews = col_double(),
##   last_review = col_date(format = ""),
##   reviews_per_month = col_double(),
##   calculated_host_listings_count = col_double(),
##   availability_365 = col_double()
## )
```

```
mydata = dplyr::select(mydata, -c(id, host_id, last_review))
mydata$reviews_per_month = replace_na(mydata$reviews_per_month, 0)
filtered = filter(mydata, price > 0 & price <= 3000 &
                    minimum_nights <= 60 & reviews_per_month <= 15)
data1 = filtered
boxcoxfit(filtered$number_of_reviews, lambda2 = TRUE)
```

```
## Fitted parameters:
##     lambda    lambda2       beta    sigmasq
##  0.1508446  0.0062900  1.6811964 11.0336866
##
## Convergence code returned by optim: 0
```

```
boxcoxfit(filtered$reviews_per_month, lambda2 = TRUE)
```

```
## Fitted parameters:
##     lambda    lambda2       beta    sigmasq
##  0.1911564  0.0001462 -1.1203662  3.9392326
##
## Convergence code returned by optim: 0
```

```
boxcoxfit(filtered$price, lambda2 = TRUE)
```

```
## Fitted parameters:
##        lambda       lambda2          beta       sigmasq
## -0.2157544671  0.0001949825  2.9438504794  0.0594732465
##
## Convergence code returned by optim: 0
```

```
boxcoxfit(filtered$minimum_nights, lambda2 = TRUE)
```

```
## Fitted parameters:
##     lambda    lambda2       beta    sigmasq
## -0.4554950  0.0000000  0.7405213  0.3079380
##
## Convergence code returned by optim: 0
```

```
boxcoxfit(filtered$calculated_host_listings_count, lambda2 = TRUE)
```

```
## Fitted parameters:
##      lambda     lambda2        beta     sigmasq
## -1.30051167  0.00000000  0.19940681  0.08230652
##
## Convergence code returned by optim: 0
```

```
test_filt = dplyr::select(filtered, -c(name, host_name, neighbourhood))
other_filt = test_filt
other_filt = other_filt %>%
  mutate(room_type =
           dplyr::recode(room_type, 'Entire home/apt' = 3,
                         'Private room' = 2,
                         'Shared room' = 1))
summary(lm(log(price) ~ ., data = other_filt))
```

```
##
## Call:
## lm(formula = log(price) ~ ., data = other_filt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0146 -0.3084 -0.0514  0.2390  3.9260
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -2.011e+02  6.818e+00 -29.495  < 2e-16 ***
## neighbourhood_groupBrooklyn     -4.126e-02  1.860e-02  -2.218   0.0266 *
## neighbourhood_groupManhattan     2.767e-01  1.688e-02  16.386  < 2e-16 ***
## neighbourhood_groupQueens        8.807e-02  1.791e-02   4.918 8.78e-07 ***
## neighbourhood_groupStaten Island -8.541e-01  3.533e-02 -24.174  < 2e-16 ***
## latitude                        -6.290e-01  6.645e-02  -9.467  < 2e-16 ***
## longitude                       -3.103e+00  7.648e-02 -40.578  < 2e-16 ***
## room_type                        7.227e-01  4.141e-03 174.532  < 2e-16 ***
## minimum_nights                  -1.051e-02  2.743e-04 -38.323  < 2e-16 ***
## number_of_reviews               -6.845e-04  6.176e-05 -11.083  < 2e-16 ***
## reviews_per_month               -1.724e-02  1.800e-03  -9.580  < 2e-16 ***
## calculated_host_listings_count   2.827e-04  7.200e-05   3.927 8.62e-05 ***
## availability_365                 9.023e-04  1.829e-05  49.332  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4811 on 48488 degrees of freedom
## Multiple R-squared:  0.5103, Adjusted R-squared:  0.5102
## F-statistic:  4211 on 12 and 48488 DF,  p-value: < 2.2e-16
```

```
summary(lm(log(price) ~ ., data = test_filt))
```

```
##
## Call:
## lm(formula = log(price) ~ ., data = test_filt)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -3.025 -0.307 -0.050  0.237  3.806
##
## Coefficients:
##                                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     -1.973e+02  6.783e+00  -29.082  < 2e-16 ***
## neighbourhood_groupBrooklyn     -2.131e-02  1.853e-02   -1.150     0.25
## neighbourhood_groupManhattan     2.863e-01  1.680e-02   17.042  < 2e-16 ***
## neighbourhood_groupQueens        1.006e-01  1.782e-02    5.646 1.65e-08 ***
## neighbourhood_groupStaten Island -8.182e-01  3.518e-02  -23.254  < 2e-16 ***
## latitude                        -5.556e-01  6.618e-02   -8.395  < 2e-16 ***
## longitude                       -3.041e+00  7.613e-02  -39.940  < 2e-16 ***
## room_typePrivate room           -7.671e-01  4.567e-03 -167.974  < 2e-16 ***
## room_typeShared room            -1.176e+00  1.453e-02  -80.919  < 2e-16 ***
## minimum_nights                  -1.061e-02  2.729e-04  -38.887  < 2e-16 ***
## number_of_reviews               -6.485e-04  6.146e-05  -10.551  < 2e-16 ***
## reviews_per_month               -1.751e-02  1.790e-03   -9.783  < 2e-16 ***
## calculated_host_listings_count   2.804e-04  7.163e-05    3.915 9.05e-05 ***
## availability_365                 8.832e-04  1.822e-05   48.484  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4786 on 48487 degrees of freedom
## Multiple R-squared:  0.5154, Adjusted R-squared:  0.5153
## F-statistic:  3967 on 13 and 48487 DF,  p-value: < 2.2e-16
```

```
summary(glm(price ~ ., data = other_filt, family = gaussian("log")))
```

```
##
## Call:
## glm(formula = price ~ ., family = gaussian("log"), data = other_filt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -376.20   -48.29   -16.55    15.89  2904.79
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -3.207e+02  1.692e+01 -18.952  < 2e-16 ***
## neighbourhood_groupBrooklyn      2.970e-03  4.922e-02   0.060   0.9519
## neighbourhood_groupManhattan     2.787e-01  4.726e-02   5.897 3.73e-09 ***
## neighbourhood_groupQueens        1.208e-01  4.998e-02   2.416   0.0157 *
## neighbourhood_groupStaten Island -1.064e+00  8.792e-02 -12.099  < 2e-16 ***
## latitude                        -2.468e-01  1.351e-01  -1.828   0.0676 .
## longitude                       -4.511e+00  1.824e-01 -24.730  < 2e-16 ***
## room_type                        7.572e-01  1.055e-02  71.769  < 2e-16 ***
## minimum_nights                  -1.414e-02  4.679e-04 -30.227  < 2e-16 ***
## number_of_reviews               -2.136e-03  1.338e-04 -15.960  < 2e-16 ***
## reviews_per_month               -3.164e-02  3.452e-03  -9.166  < 2e-16 ***
## calculated_host_listings_count  -7.235e-04  8.152e-05  -8.875  < 2e-16 ***
## availability_365                 1.535e-03  2.903e-05  52.897  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 18466.02)
##
```

```
##     Null deviance: 1167630525  on 48500  degrees of freedom
## Residual deviance:  895376733  on 48488  degrees of freedom
## AIC: 614113
##
## Number of Fisher Scoring iterations: 8
```

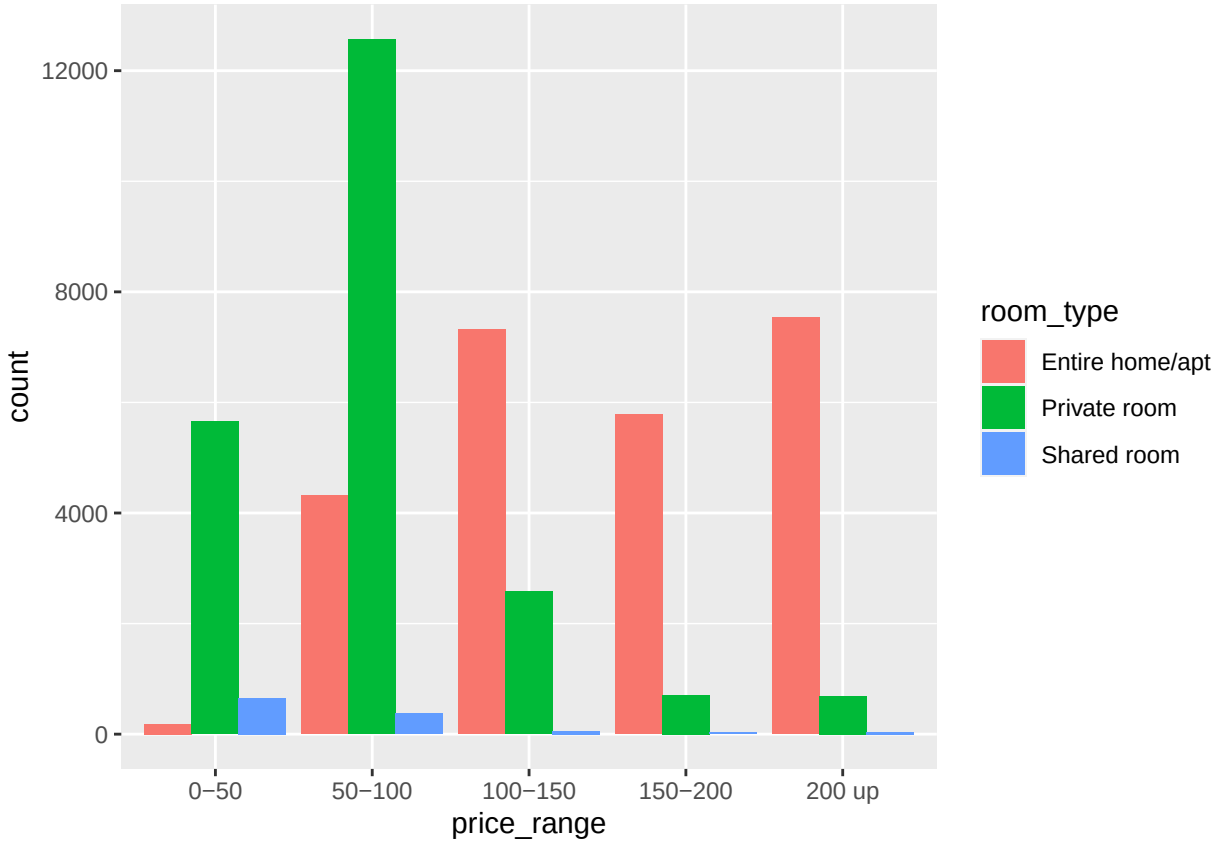
```
summary(glm(price ~ ., data = test_filt, family = gaussian("log")))
```

```
##
## Call:
## glm(formula = price ~ ., family = gaussian("log"), data = test_filt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -376.04   -48.32   -16.56    15.73  2906.05
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -3.185e+02  1.691e+01 -18.834  < 2e-16 ***
## neighbourhood_groupBrooklyn      4.449e-03  4.912e-02   0.091   0.9278
## neighbourhood_groupManhattan     2.784e-01  4.716e-02   5.903 3.59e-09 ***
## neighbourhood_groupQueens        1.222e-01  4.988e-02   2.451   0.0143 *
## neighbourhood_groupStaten Island -1.058e+00  8.778e-02 -12.047  < 2e-16 ***
## latitude                        -2.317e-01  1.350e-01  -1.717   0.0860 .
## longitude                       -4.504e+00  1.823e-01 -24.705  < 2e-16 ***
## room_typePrivate room           -7.709e-01  1.096e-02 -70.339  < 2e-16 ***
## room_typeShared room            -1.164e+00  5.747e-02 -20.257  < 2e-16 ***
## minimum_nights                  -1.415e-02  4.679e-04 -30.254  < 2e-16 ***
## number_of_reviews               -2.126e-03  1.338e-04 -15.892  < 2e-16 ***
## reviews_per_month               -3.189e-02  3.452e-03  -9.236  < 2e-16 ***
## calculated_host_listings_count  -7.203e-04  8.151e-05  -8.836  < 2e-16 ***
## availability_365                 1.531e-03  2.903e-05  52.731  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 18455.58)
##
##     Null deviance: 1167630525  on 48500  degrees of freedom
## Residual deviance:  894852964  on 48487  degrees of freedom
## AIC: 614087
##
## Number of Fisher Scoring iterations: 8
```

Begin Analysis

```
data1 = data1 %>%
  mutate(price_range =
           cut(price, breaks = c(0,50,100,150,200,99999),
               labels = c("0-50","50-100","100-150","150-200","200 up")))
```

```
ggplot(data1, aes(price_range)) + geom_bar(aes(fill = room_type), position = "dodge")
```

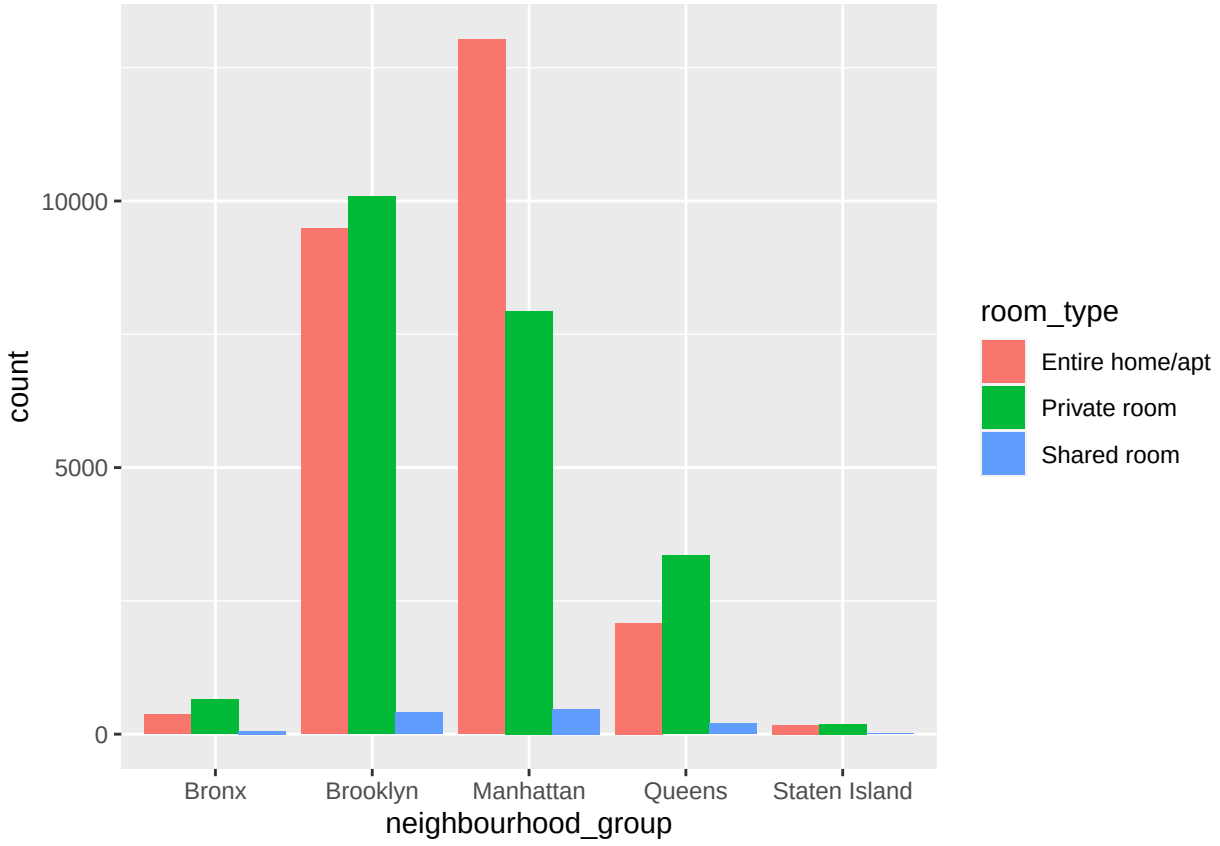



```
# room type and their avg price
data1 %>%
  group_by(room_type) %>%
  summarise(u = mean(price))
```

```
## # A tibble: 3 x 2
##   room_type           u
## * <chr>           <dbl>
## 1 Entire home/apt 204.
## 2 Private room     87.4
## 3 Shared room      70.5
```

proof of the entire room is more expensive. (graph + avg price)

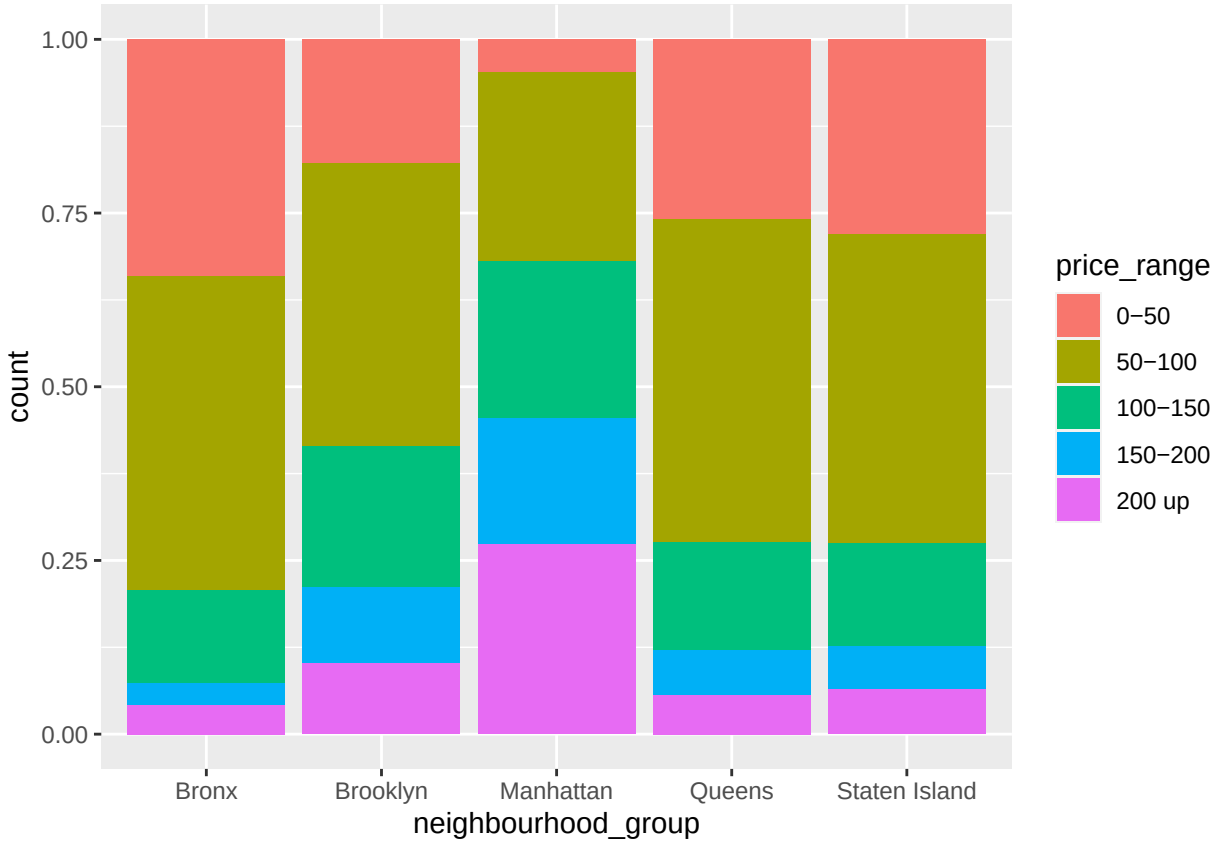
```
# histogram of city and room type
ggplot(data1, aes(neighbourhood_group)) + geom_bar(aes(fill = room_type), position = "dodge")
```



Entire home and private room much more popular than shared room. People want privacy. Most boroughts have more private rooms than entire room, but not in Manhattan. Reasonable because Manhattan is the richest boroughs in NYC

https://nypost.com/2019/12/12/gdp-in-nycs-outer-boroughs-leads-state-in-economic-output/

```
ggplot(data1, aes(neighbourhood_group)) + geom_bar(aes(fill = price_range), position = "fill")
```

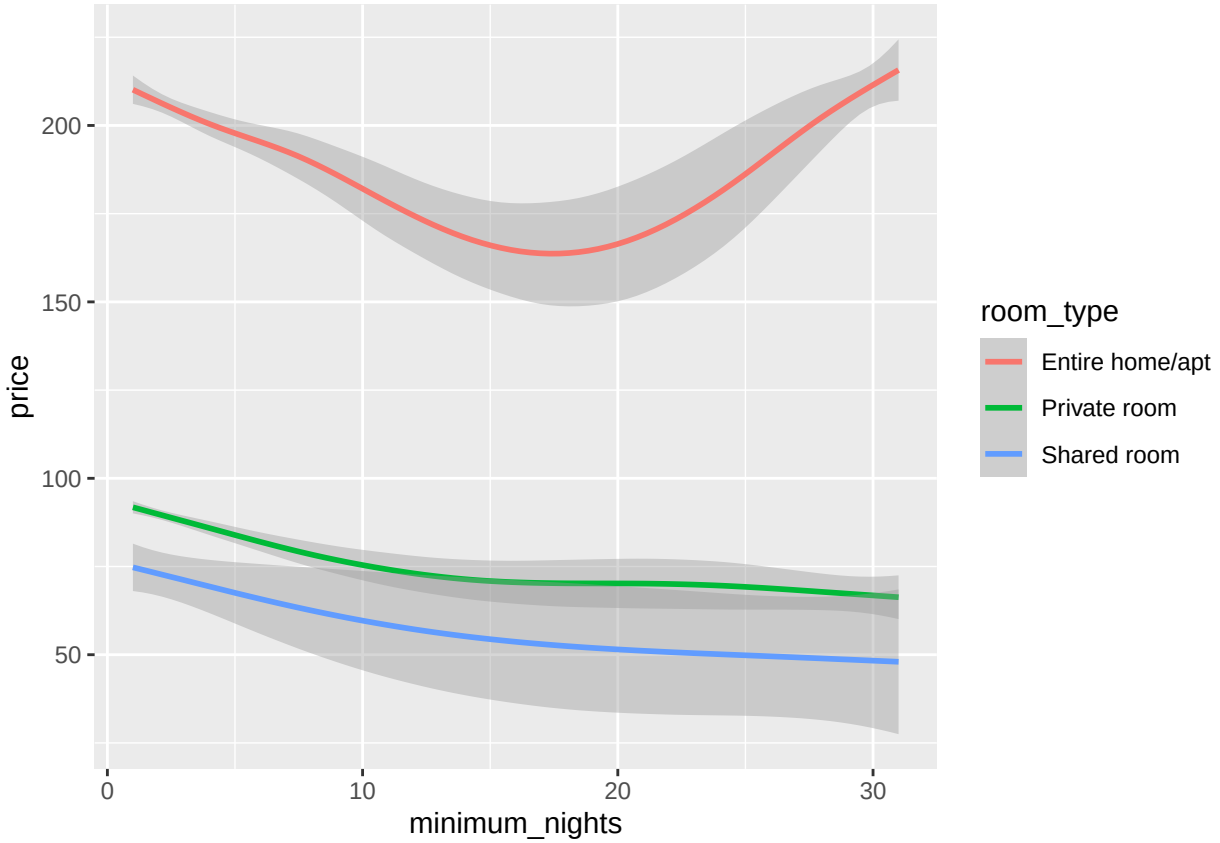


Combine previous 2 graph's information, get distribution

what affects price?

```
# get short term rental data
# minimum nights <= 31 (one month)
mn31 = data1 %>%
  filter(minimum_nights <= 31)
# plot against price and room type
ggplot(mn31, aes(minimum_nights, price)) + geom_smooth(aes(color = room_type))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

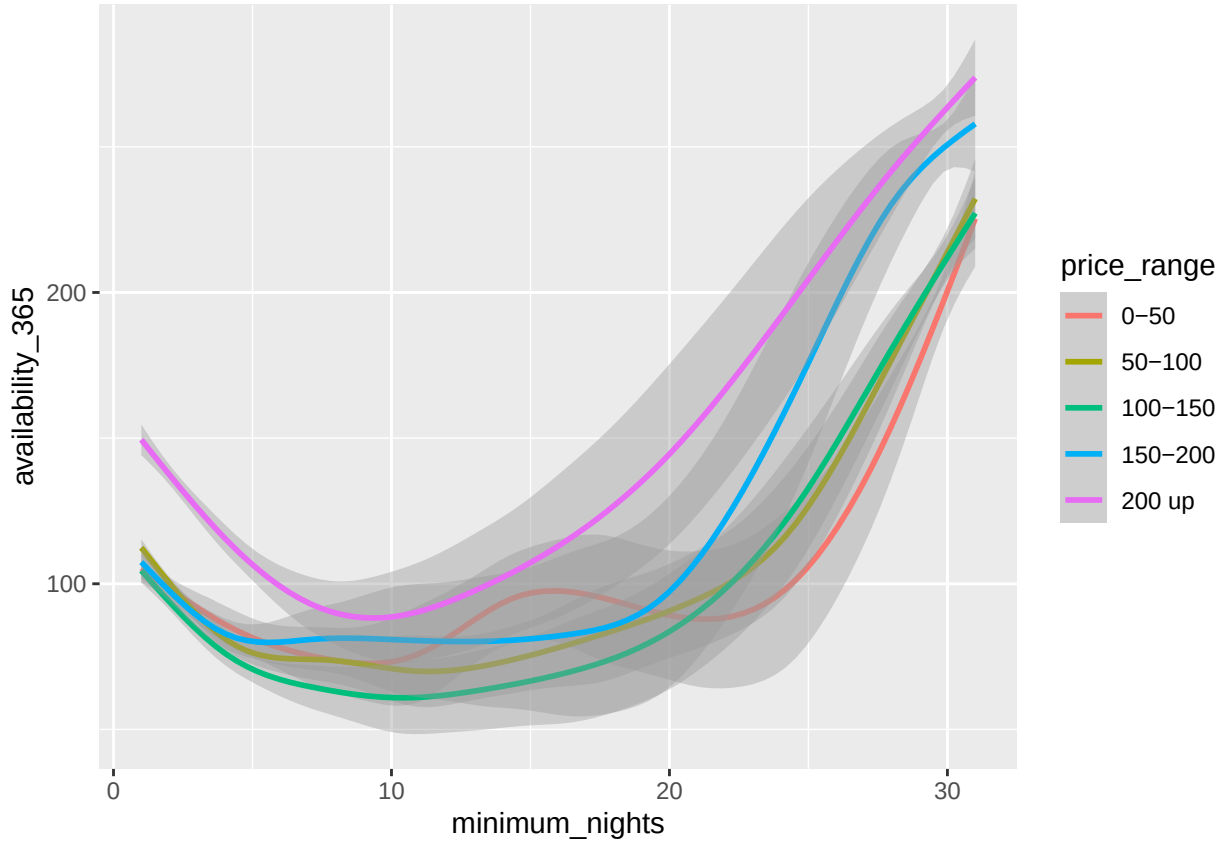


Private and shared rooms' prices drop as minimum night increases. This fits economic rule (like risk premium?), the longer stay, the lower per night price.

Entire room's price decrease first, then start increasing at mn=20. Maybe longer stay means the room is better? I'm not sure.

```
a50 = data1 %>%
  filter(availability_365 <= 50 & minimum_nights <= 31)
ggplot(mn31, aes(minimum_nights, availability_365)) + geom_smooth(aes(col = price_range))
```

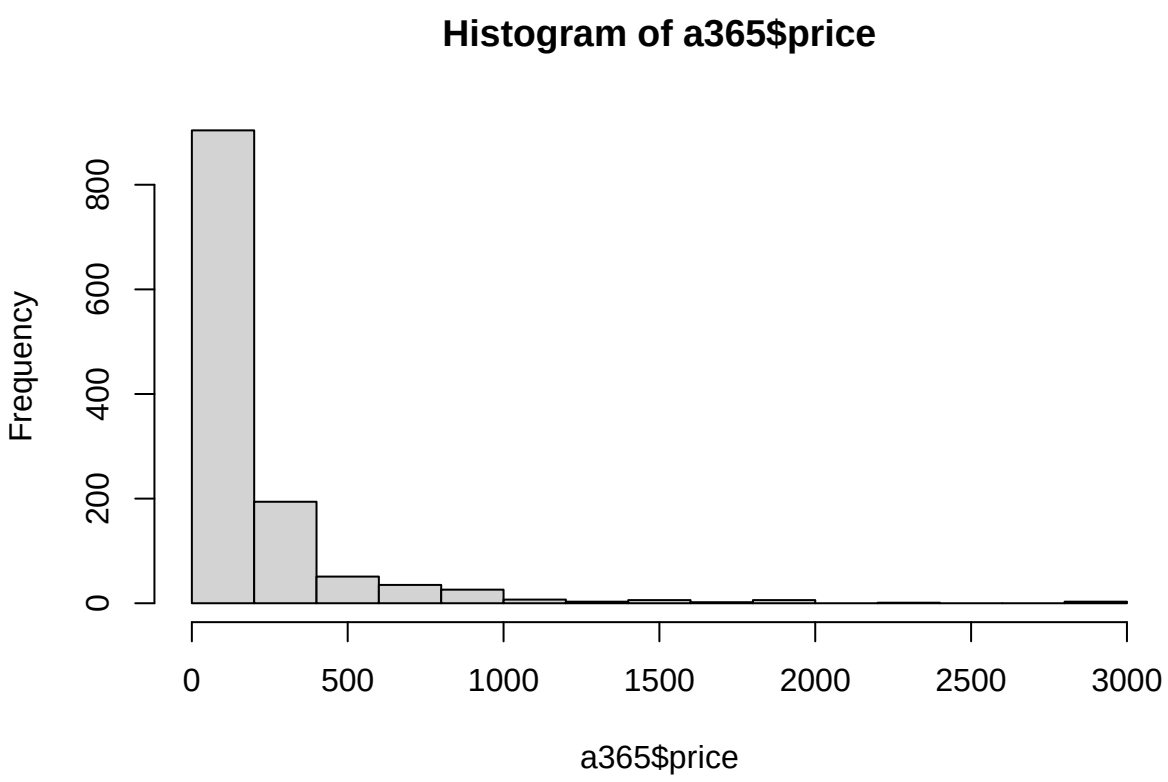
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Overall, availability would increase as minimum nights increase. This fits our expectation because people who rent Airbnb prefer stay short in a room (people can just rent a house/apartment if they have to stay long). Or people tend to rent Airbnb during vacation, which is around a week or two.

Just some sketches, you can ignore it.

```
a365 = data1 %>%
  filter(availability_365 == 365)
hist(a365$price)
```




```
data1 %>%
  group_by(neighbourhood_group, price_range) %>%
  summarize(u = mean(availability_365))
```

```
## `summarise()` has grouped output by 'neighbourhood_group'. You can override using the `.groups` argu
```

```
## # A tibble: 25 x 3
## # Groups:   neighbourhood_group [5]
##    neighbourhood_group price_range     u
##    <chr>               <fct>       <dbl>
##  1 Bronx               0-50        151.
##  2 Bronx               50-100      169.
##  3 Bronx               100-150     169.
##  4 Bronx               150-200     154.
##  5 Bronx               200 up      235.
##  6 Brooklyn            0-50         92.9
##  7 Brooklyn            50-100       94.7
##  8 Brooklyn            100-150      97.0
##  9 Brooklyn            150-200     111.
## 10 Brooklyn            200 up      125.
## # ... with 15 more rows
```

```
data1 %>%
  filter(availability_365 <= 50) %>%
  group_by(neighbourhood_group, price_range) %>%
  count()
```

```
## # A tibble: 25 x 3
```

```
## # Groups:   neighbourhood_group, price_range [25]
##    neighbourhood_group price_range     n
##    <chr>               <fct>       <int>
##  1 Bronx               0-50          116
##  2 Bronx               50-100        132
##  3 Bronx               100-150        40
##  4 Bronx               150-200        11
##  5 Bronx               200 up          9
##  6 Brooklyn            0-50         2155
##  7 Brooklyn            50-100       4626
##  8 Brooklyn            100-150      2217
##  9 Brooklyn            150-200      1117
## 10 Brooklyn            200 up        914
## # ... with 15 more rows
```