Making fRiends with: ggplot2

Pamela Matias

June 21st 2017

Outline

- Installing R and R Studio
- Getting workshop files & data
  - Download/Clone from GitHub
- Nice to meet you, ggplot2
- Dataset: AirBnB in Berlin
- Basic types of plots
- Exercises

This is an R Markdown script. When you execute code within the notebook, the results appear beneath the code ;)

You can go executing different code chunks individually by clicking the *Run* button at the top of the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*. You can also just run a single line if you highlight it and then press *Cmd+Enter*.

If you don't feel so comfortable working with RMarkdown, check out the available .R script in our workshop files - that you can just work with normally.

Installing R and RStudio

Open a web browser and go to http://cran.r-project.org, then download and install it

RStudio would also be helpful (get at http://rstudio.com)

In R, install the ggplot2 package:

```
#install.packages("ggplot2")
```

Getting workshop files & data

All of the files are to be found here 20170521__2nd__meetup__ggplot2intro

You can clone the repo or simply download the files :)

Nice to meet you, ggplot2

So we are officially taking our first steps with Hardley Wickhams package ggplot2. Check out this website for more detailed information.

What are we aiming for here? In the next 10 to 15 minutes we will understand the underlying grammar of ggplot2, and explore how to code and modify five basic plot types this package has to offer.

Make sure to check out the official, updated info for the [ggplot2 package] (https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf)

Warm-up questions What can one do with ggplot and why use it? * ready-to-publish plots * plot design at a high level of abstraction * very flexible * systematic structuring for polishing plot appearance * mature and complete graphics system * many users, active mailing list

What can't I do with ggplot2? * 3-dimensional graphics (see the rgl package) * Graph-theory type graphs (nodes/edges layout; see the igraph package) * Interactive graphics (see the ggvis package)

Dataset: AirBnB in Berlin!

To do something slightly different from the classic iris and diamonds, today we'll be packing our bags and off to Berlin we go!

Or at least we'll do so mentally, since we'll be taking a look at data from AirBnB in this German city (sorry, there was no data available for our beloved Munich #muchdisappointment).

Anyway, Inside Airbnb is a project that uses publicly available information from the Airbnb web-site. It includes the availability calendar for 365 days in the future, and the reviews for each listing. Since every detail is taken from what is out there, there is no "private" information being used without people's consents - all of the host and place information is already available online.

A lot of work has been put into verifying, cleaning, aggregating and analyzing the data by an entire team, aand it's been made available below under a Creative Commons CC0 1.0 Universal (CC0 1.0) "Public Domain Dedication" license - so in case you're interested, you can also download and play around with tons and tons of data from several other cities.

Btw, the platform offers an online visualization tool. Our ggplots will be cool enough, but it could be worth taking a look at this tool.

Now, let's get to work. We need to download the data and read it in R; this first chunk will make our lifes easier when importing it directly from GitHub:

```
#install.packages("RCurl")
library(RCurl)
```

```
## Warning: package 'RCurl' was built under R version 3.2.4
```

```
## Loading required package: bitops
```

```
library(RColorBrewer)
```

Note: There are two datasets online:

- The raw data, as directly downloaded from Inside Airbnb:
- Our workshop data, which has some minor corrections for funny characters appearing in German names (such as the emoji looking o's and u's *mit umlaut*)

```
# Option 1: Importing directly from GitHub using RCurl
x <- getURL("https://raw.githubusercontent.com/pamelamatias/RLadies_MUC/master/20170521_2nd_meetup_ggpl
berlin <- read.csv(text = x)

#Option 2: Download and save (or clone) data, then just type in the corresponding directory and off we
berlin<-read.table("/Users/Pam/Documents/GitHub/rladies_munich/20170521_2nd_meetup_ggplot2intro/worksho

# Check nothing funny happened while we were importing it
head(berlin)
```

```
# Option 1: Importing directly from GitHub using RCurl
x <- getURL("https://raw.githubusercontent.com/pamelamatias/RLadies_MUC/master/20170521_2nd_meetup_ggpl
berlin <- read.csv(text = x)

# Check nothing funny happened while we were importing it
head(berlin)
```

```
##   X       id   host_id host_name neighbourhood_group
## 1 1 17260587 110413588     Julia              Pankow
## 2 2 17227881 116030782   Jessica              Pankow
## 3 3  1153584   4813229    Ulrike              Pankow
## 4 4  7832750  36417780      Lisa              Pankow
## 5 5 11634962  12101514   Elektra              Pankow
## 6 6 13035058  21797784  Jennifer              Pankow
##                          neighbourhood latitude longitude    room_type
## 1 Blankenfelde/Niedersch<U+00F6>nhausen 52.58331  13.39609 Private room
## 2 Blankenfelde/Niedersch<U+00F6>nhausen 52.57642  13.39858 Private room
## 3 Blankenfelde/Niedersch<U+00F6>nhausen 52.58922  13.39926 Private room
## 4 Blankenfelde/Niedersch<U+00F6>nhausen 52.58231  13.39558 Private room
## 5 Blankenfelde/Niedersch<U+00F6>nhausen 52.58375  13.39366 Private room
## 6 Blankenfelde/Niedersch<U+00F6>nhausen 52.57967  13.39932 Private room
##   price minimum_nights number_of_reviews last_review reviews_per_month
## 1    30              2                 2  2017-05-03              2.00
## 2    17              1                 3  2017-03-22              1.29
## 3    50              2                56  2017-04-16              1.27
## 4    28              2               111  2017-04-25              5.43
## 5    21              1                17  2017-05-05              1.45
## 6    38              2                19  2016-12-26              1.71
##   calculated_host_listings_count availability_365
## 1                              3               44
## 2                              1               36
## 3                              1              124
## 4                              1                7
## 5                              1               64
## 6                              1              185
```

As you can see, `berlin` has >20k listings for AirBnB places in Berlin, and 15 observations per listing. This dataset is actually a snapshot of the listings available at a particular time (in this case, up to last month). Let's take a quick look at the variables:

```r
#Define type of variable
berlin$id <- as.integer(berlin$id)
berlin$host_id <- as.integer(berlin$host_id)
berlin$host_name <- as.factor(berlin$host_name)
berlin$neighbourhood_group <- as.factor(berlin$neighbourhood_group)
berlin$neighbourhood <- as.factor(berlin$neighbourhood)
berlin$room_type <- as.factor(berlin$room_type)
berlin$number_of_reviews <- as.numeric(berlin$number_of_reviews)
berlin$last_review <- as.Date(berlin$last_review)
berlin$calculated_host_listings_count <- as.numeric(berlin$calculated_host_listings_count)
berlin$availability_365 <- as.numeric(berlin$availability_365)
berlin$minimum_nights <- as.numeric(berlin$minimum_nights)
berlin$reviews_per_month <- as.numeric(berlin$reviews_per_month)
berlin$price <- as.numeric(berlin$price)
berlin <- berlin[,-1]
#And check out its structure
dim(berlin) #20576 x 16
```

```
## [1] 20576    15
```

```r
str(berlin)
```

```
## 'data.frame':    20576 obs. of  15 variables:
##  $ id                            : int  17260587 17227881 1153584 7832750 11634962 13035058 16601732
##  $ host_id                       : int  110413588 116030782 4813229 36417780 12101514 21797784 66145
##  $ host_name                     : Factor w/ 5624 levels "","(EMAIL HIDDEN)",..: 2727 2565 5274 3195
##  $ neighbourhood_group           : Factor w/ 12 levels "Charlottenburg-Wilm.",..: 7 7 7 7 7 7 7 7 7
##  $ neighbourhood                 : Factor w/ 137 levels "Adlershof","Albrechtstr.",..: 15 15 15 15 1
##  $ latitude                      : num  52.6 52.6 52.6 52.6 52.6 ...
##  $ longitude                     : num  13.4 13.4 13.4 13.4 13.4 ...
##  $ room_type                     : Factor w/ 3 levels "Entire home/apt",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ price                         : num  30 17 50 28 21 38 40 25 37 30 ...
##  $ minimum_nights                : num  2 1 2 2 1 2 1 14 4 1 ...
##  $ number_of_reviews             : num  2 3 56 111 17 19 0 0 4 0 ...
##  $ last_review                   : Date, format: "2017-05-03" "2017-03-22" ...
##  $ reviews_per_month             : num  2 1.29 1.27 5.43 1.45 1.71 NA NA 0.09 NA ...
##  $ calculated_host_listings_count: num  3 1 1 1 1 1 1 1 1 1 ...
##  $ availability_365              : num  44 36 124 7 64 185 88 0 88 0 ...
```

```
#summary() of each variable: five-point statistics for numeric variables, while number of entries per l

summary(berlin)
```

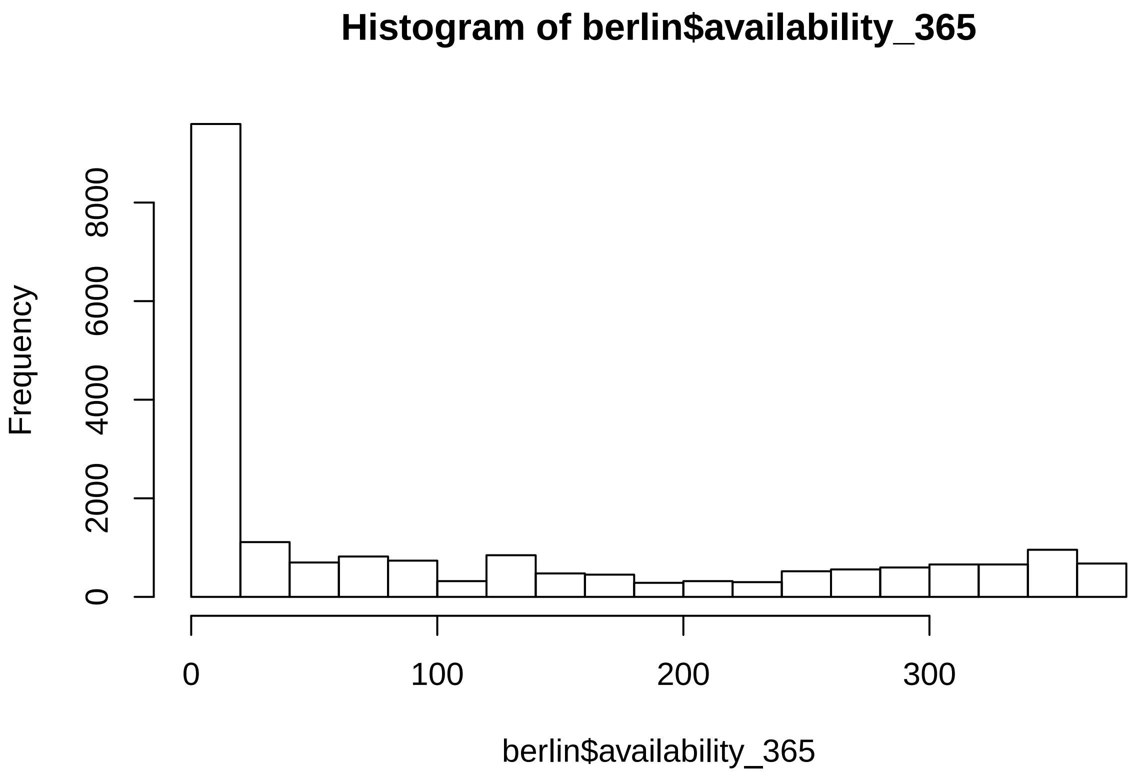
```
##        id              host_id             host_name    
##  Min.   :    1971   Min.   :     2164   Anna   :  192  
##  1st Qu.: 5544077   1st Qu.:  7338631   Julia  :  179  
##  Median :10156210   Median : 22234848   Daniel :  130  
##  Mean   :10137757   Mean   : 33711186   Martin :  123  
##  3rd Qu.:15230135   3rd Qu.: 48585570   Thomas :  120  
##  Max.   :18614499   Max.   :129212521   David  :  115  
##                                         (Other):19717  
##               neighbourhood_group                         neighbourhood  
##  Friedrichshain-Kreuzberg:5082     Tempelhofer Vorstadt          : 1220  
##  Mitte                   :4106     Frankfurter Allee S<U+00FC>d FK: 1218  
##  Neukoelln               :3461     Reuterstra<U+00DF>e            : 1059  
##  Pankow                  :3363     Alexanderplatz                :  969  
##  Charlottenburg-Wilm.    :1346     Rixdorf                       :  846  
##  Tempelhof - Schoeneberg :1343     Brunnenstr. S<U+00FC>d         :  745  
##  (Other)                 :1875     (Other)                       :14519  
##     latitude       longitude                 room_type         price        
##  Min.   :52.35   Min.   :13.12   Entire home/apt:10285   Min.   :   0.00  
##  1st Qu.:52.49   1st Qu.:13.38   Private room   :10011   1st Qu.:  30.00  
##  Median :52.51   Median :13.42   Shared room    :  280   Median :  45.00  
##  Mean   :52.51   Mean   :13.41                           Mean   :  58.28  
##  3rd Qu.:52.53   3rd Qu.:13.44                           3rd Qu.:  69.00  
##  Max.   :52.65   Max.   :13.74                           Max.   :9000.00  
##                                                                          
##  minimum_nights    number_of_reviews  last_review        
##  Min.   :  1.000   Min.   :  0.00    Min.   :2011-01-26  
##  1st Qu.:  1.000   1st Qu.:  1.00    1st Qu.:2016-08-18  
##  Median :  2.000   Median :  3.00    Median :2017-02-26  
##  Mean   :  4.451   Mean   : 12.91    Mean   :2016-11-14  
##  3rd Qu.:  3.000   3rd Qu.: 12.00    3rd Qu.:2017-04-22  
##  Max.   :360.000   Max.   :360.00    Max.   :2017-05-08  
##  NA's   :3                           NA's   :4396        
##  reviews_per_month calculated_host_listings_count availability_365
##  Min.   : 0.010    Min.   : 1.000                 Min.   :  0.0   
##  1st Qu.: 0.220    1st Qu.: 1.000                 1st Qu.:  0.0   
##  Median : 0.570    Median : 1.000                 Median : 33.0   
##  Mean   : 1.065    Mean   : 1.698                 Mean   :104.7   
##  3rd Qu.: 1.380    3rd Qu.: 1.000                 3rd Qu.:211.0   
##  Max.   :10.620    Max.   :34.000                 Max.   :365.0   
##  NA's   :4396
```

BaseGraphics vs ggplot2

Compared to base graphics, ggplot2: * can be more verbose for simple plots * is definitely less verbose for complex and custom graphics * requires data to be in a data.frame * uses a different system for adding plot elements as layers

Let's take another look at the example we had before for the BaseGraphics:

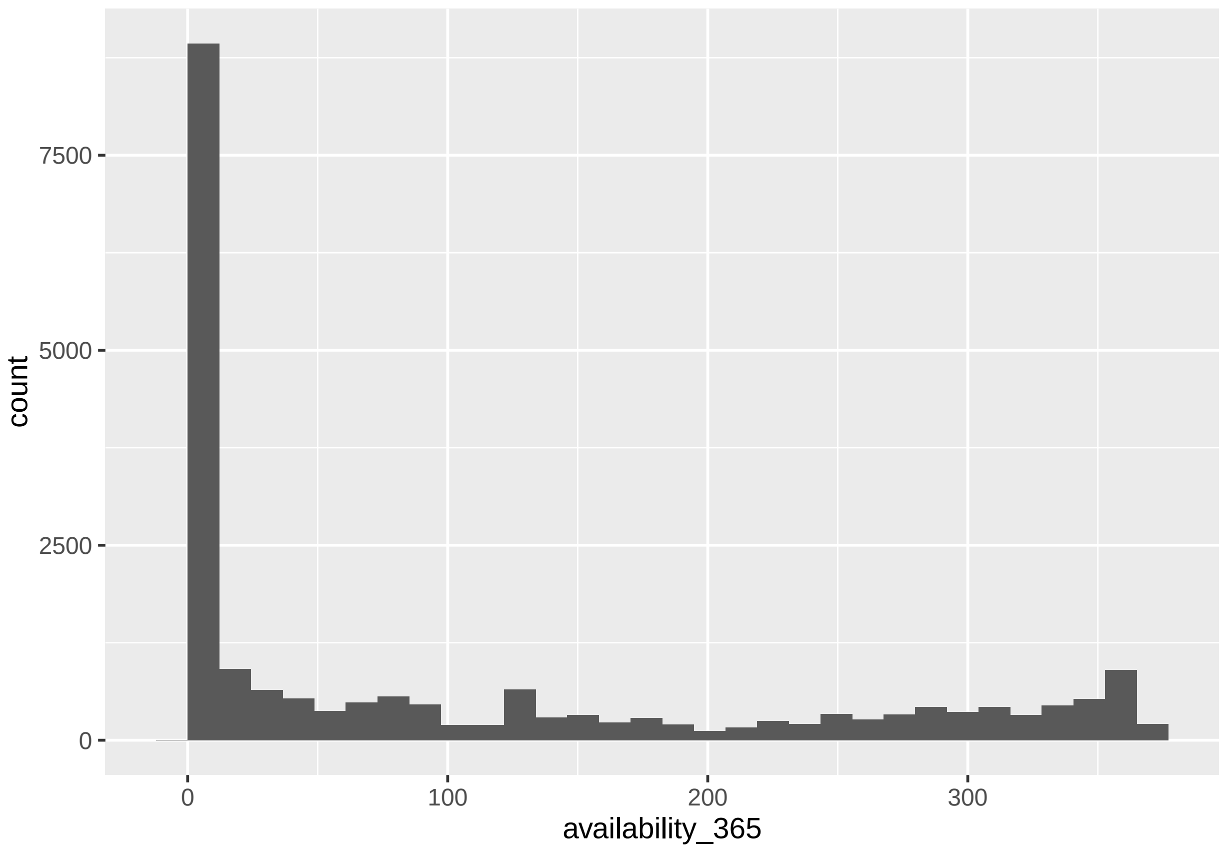
```r
hist(berlin$availability_365)
```



and same example with ggplot2:

```r
library(ggplot2)
ggplot(berlin, aes(x = availability_365)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



But what if we want to do something slightly more complex?

That's where the magic of **ggplot2** comes in handy ;)

Aesthetic mapping and geometric objects

Aesthetics mapping stands for the parameters that we can actually see in our plots. * position (i.e., on the x and y axes) * color ("outside" color) * fill ("inside" color) * shape (of points) * linetype * size These parameters are set with the `aes()` function. Of note, each type of geom accepts only a specific subset of all aesthetics. Check out the the geom help pages to see what mappings each geom accepts.

Geometric objects are the actual marks that we put on a plot, such as points, lines or boxplots. They are encoded in this package as: * points: `geom_point()`, useful for scatter plots and dot plots. * lines: `geom_line()`, for line plots as time series and trend lines. * bar charts and histograms: `geom_bar()`, `geom_histogram()` * boxplots: `geom_boxplot()` or `stat_boxplot()` * multiple facets of variables: using `facet_grid()` or `facet_wrap()`

And these geom_xy() are usually added as a second step to the definition of our data space.

From here, all we need to get started is defining our dataset and the variables we wish to see plotted on the x- and y-axis, and gradually build on this.

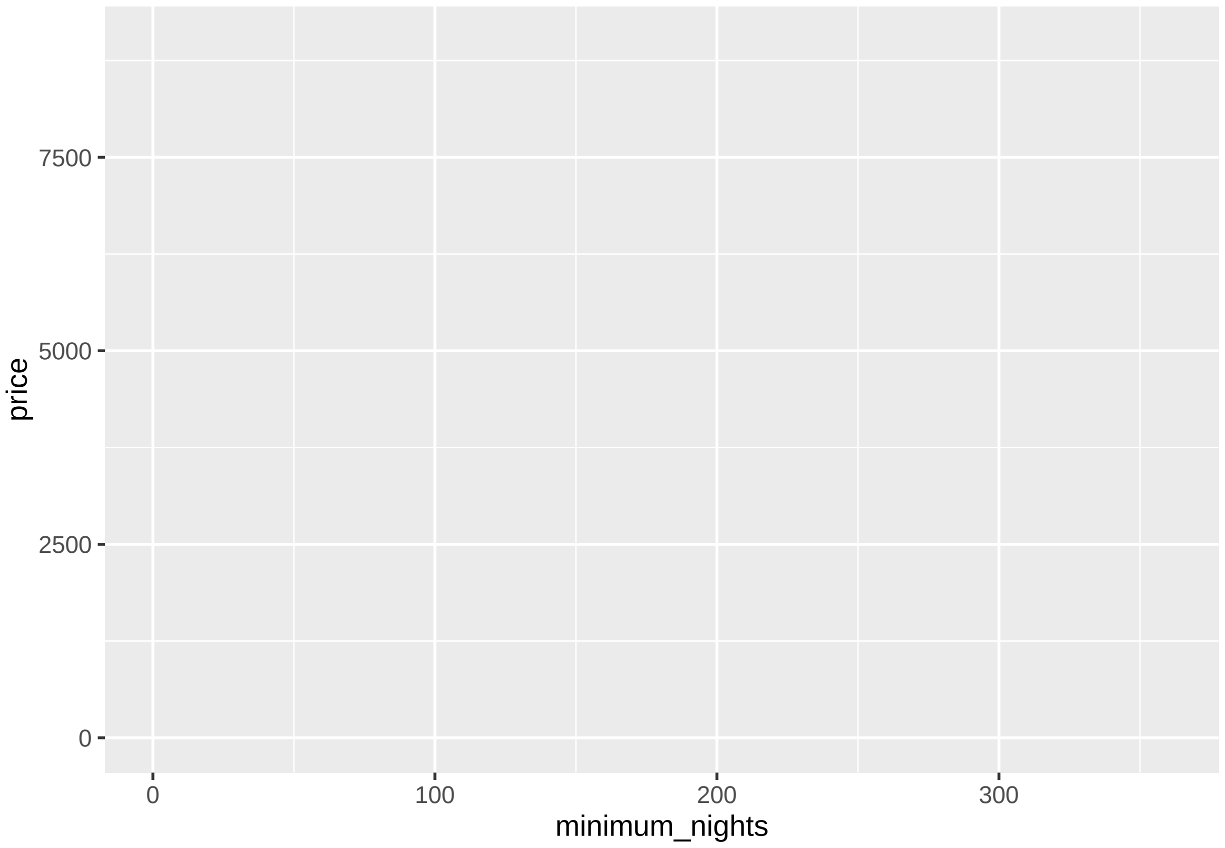
Basic plots

1 - Scatter plot

Scatter plots are useful when comparing two continous variables

Minimum length of stay vs price Let's explore the relationship between the minimum stay nights and the price per night.

```
#definition of the data set and variables for each axis.
ggplot(berlin, aes(x = minimum_nights, y = price))
```



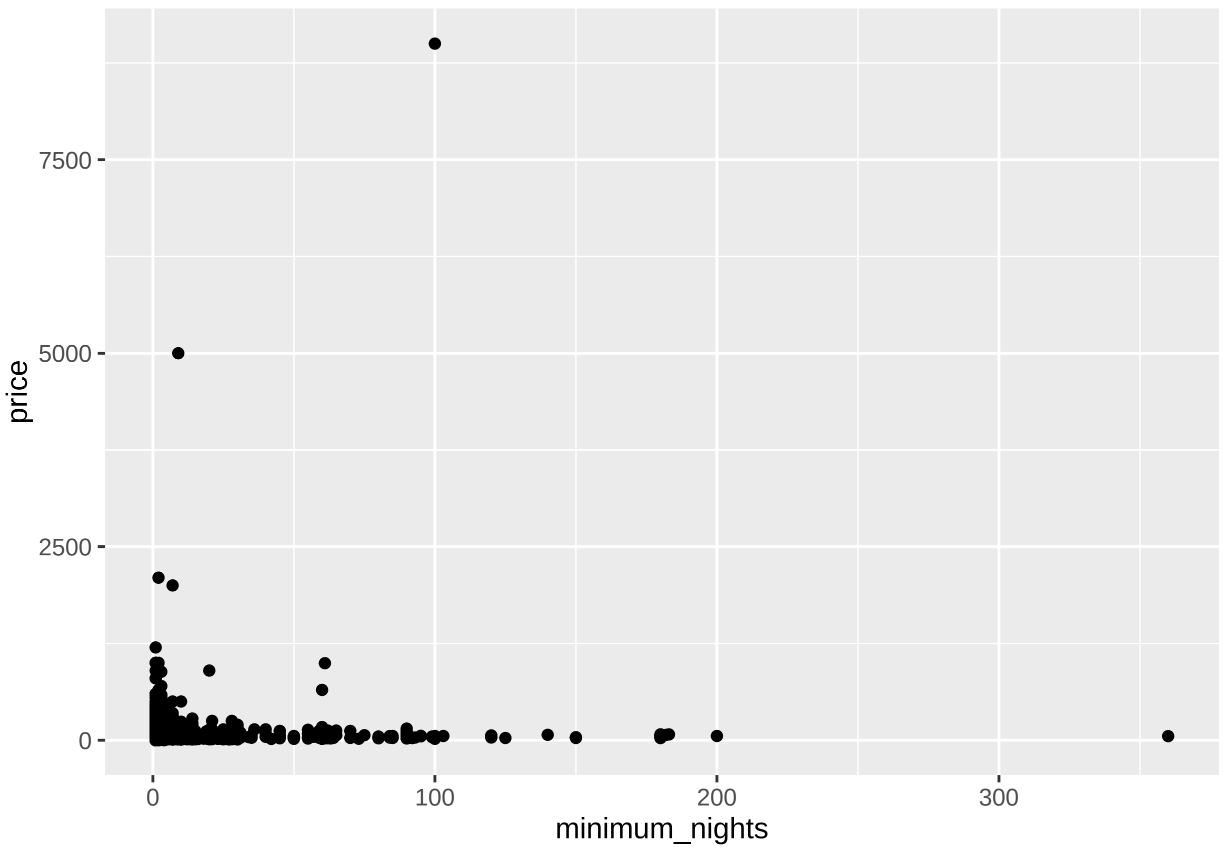
Wait a second, no plot was created. Why?

Remember, we need to define how will ggplot2 actually plot our information and with which type of geometric object **geom_xy()**.

The **geom_point()** creates a dot for each data point corresponding to the data.

```
#adding geom_point()
ggplot(berlin, aes(x = minimum_nights, y = price)) +
  geom_point()
```

```
## Warning: Removed 3 rows containing missing values (geom_point).
```

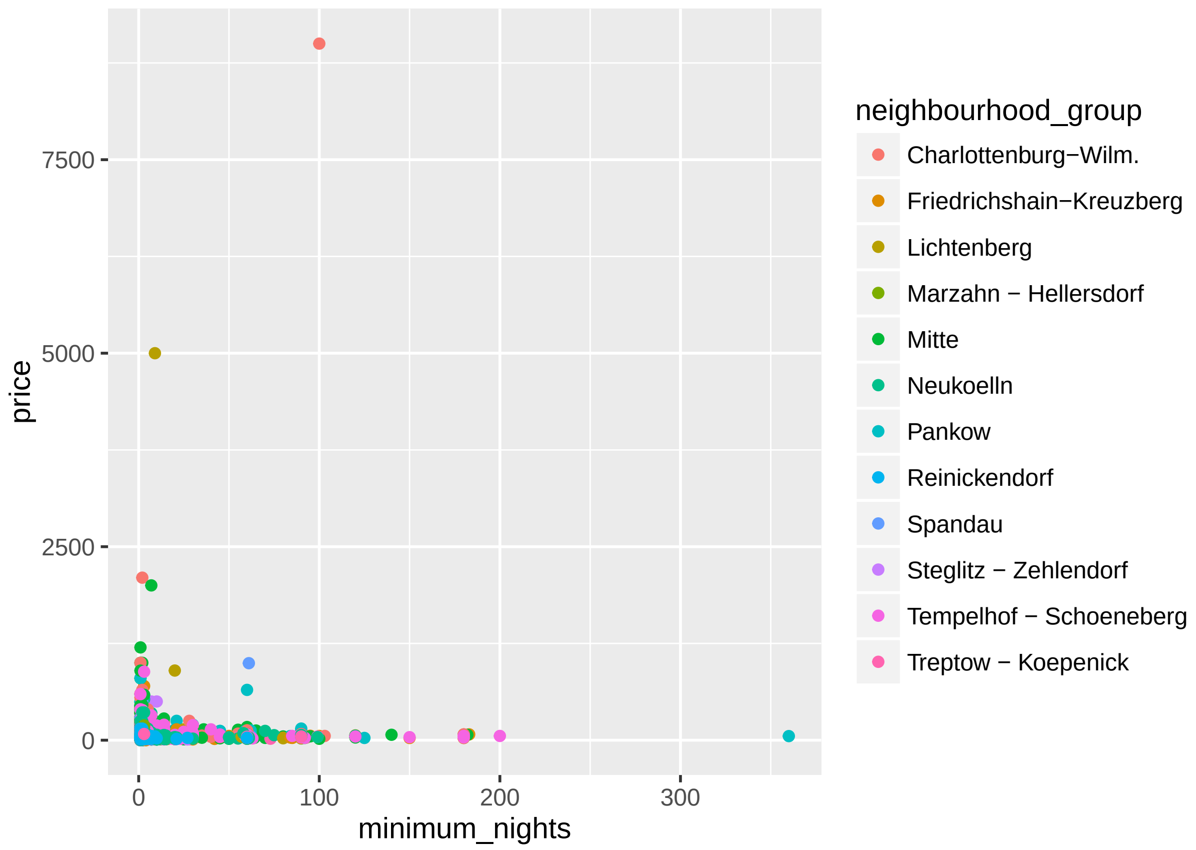


Minimum length of stay vs price What if we add another variable that could be potentially related to the price? The physical location of a BnB home could affect the pricing, but how could our plot look if we actually identified *neighbourhood* group?

In order to color code for each dot in our plot according to the neighbourhood it corresponds to, we have to define an aesthetics for each data point in the **geom_point()** as shown below in the code.

```r
#Identify the neighbourhood group to which each dot belongs by color coding for it
ggplot(berlin, aes(x = minimum_nights, y = price)) +
  geom_point(aes(color = neighbourhood_group))
```

```
## Warning: Removed 3 rows containing missing values (geom_point).
```



Unless it's a typo in our data, I think I *don't really* want to stay in that home in Charlottenburg-Wilm: for more than 7500EUR a night, with a minimum stay of around 100 days? No thanks.

---

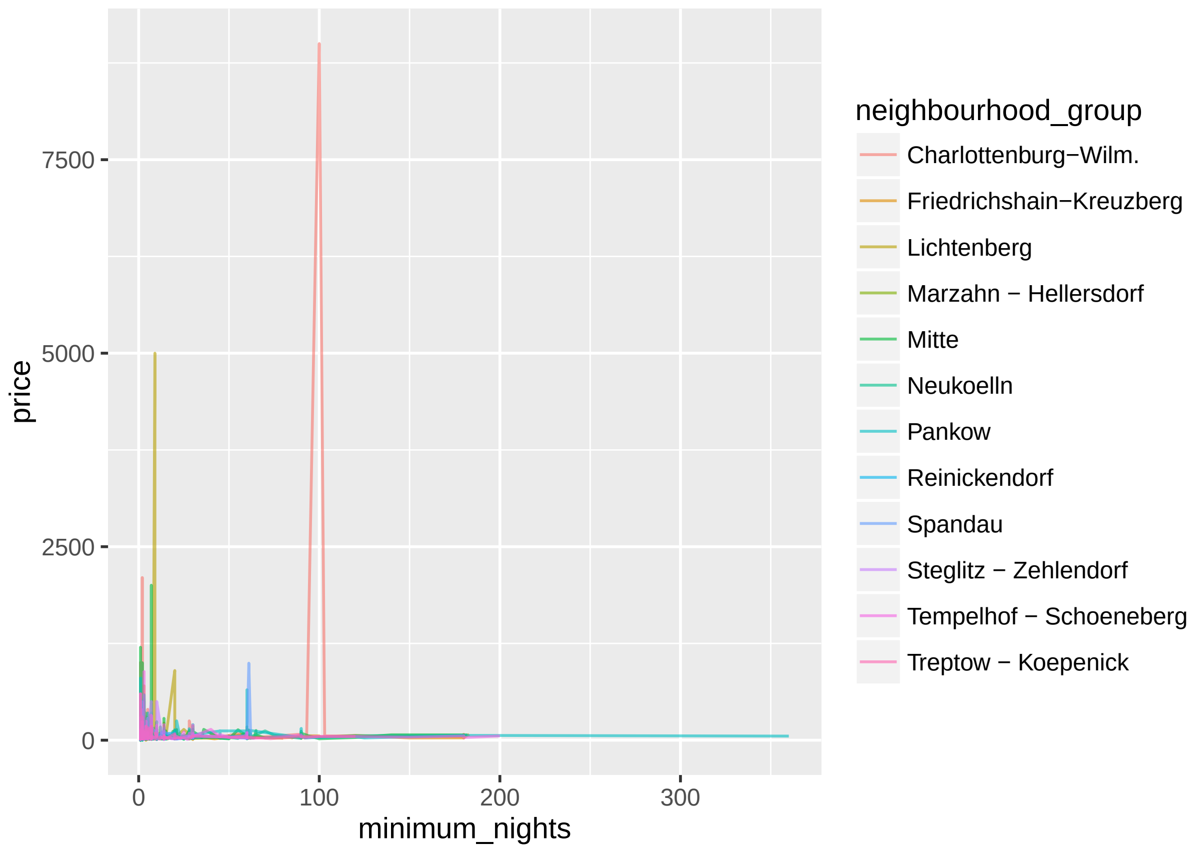
2 - Line plot

Line plots are useful for the visualizing a discrete (x-axis) and a continous (y-axis) variable.

Minimum stay vs price With this chunk we're doing a similar plot to the one we had before, but this time as a line plot, and playing with the transparency within the **geom_line()** parameter.

```
#line plots and alpha (transparency)
ggplot(berlin, aes(x=minimum_nights, y=price, color=neighbourhood_group)) +
    geom_line(alpha=0.6)
```

```
## Warning: Removed 3 rows containing missing values (geom_path).
```



3 - Histograms and bar charts

What about 1D visualization of either categorical or continuous data?

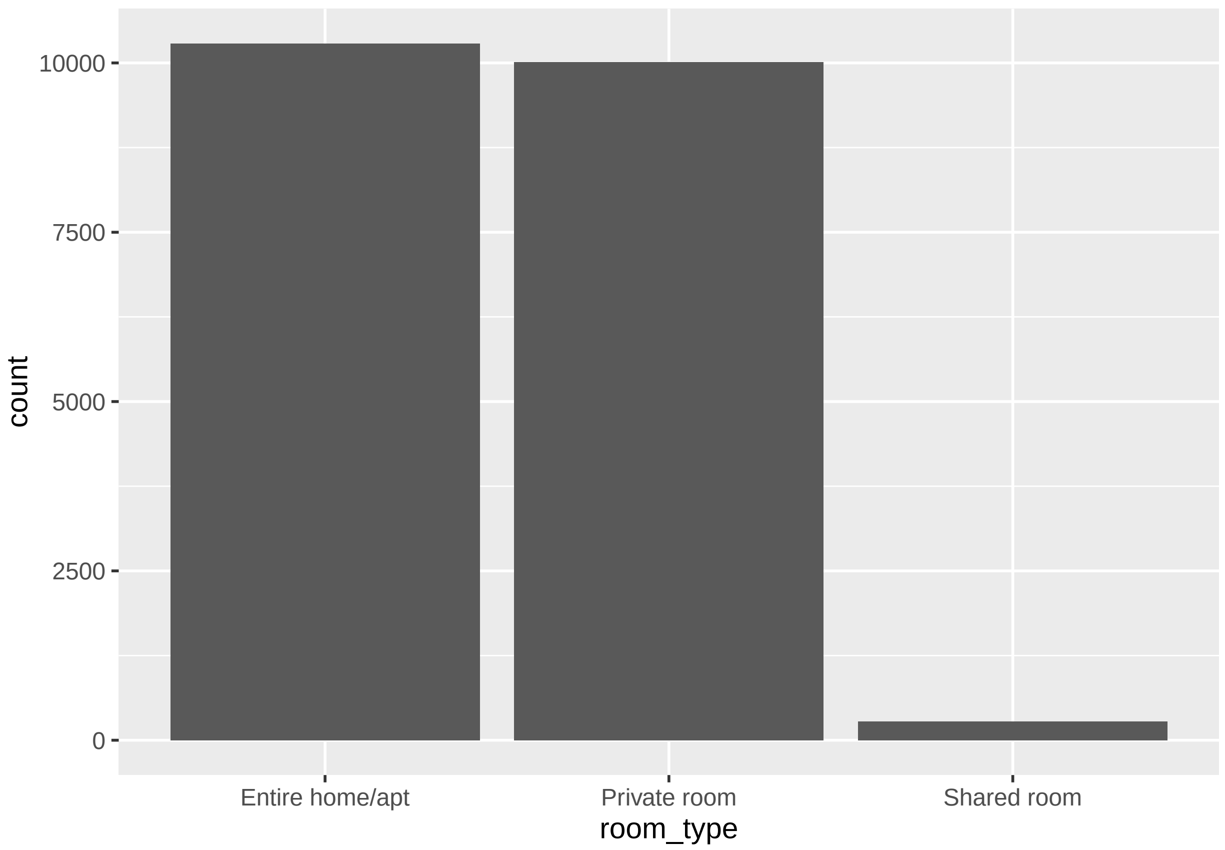
geom_bar() and **geom_histogram()** do the job for cases in which a categorical variable is defined for the x axis, while a continuous one is done so for the y axis.

Is is only necessary to specify the variable x when coding for our plot, since this is a simple visualization of a single variable.

Bar chart Here a nice example for the number of listings (or homes) per type of room offered.

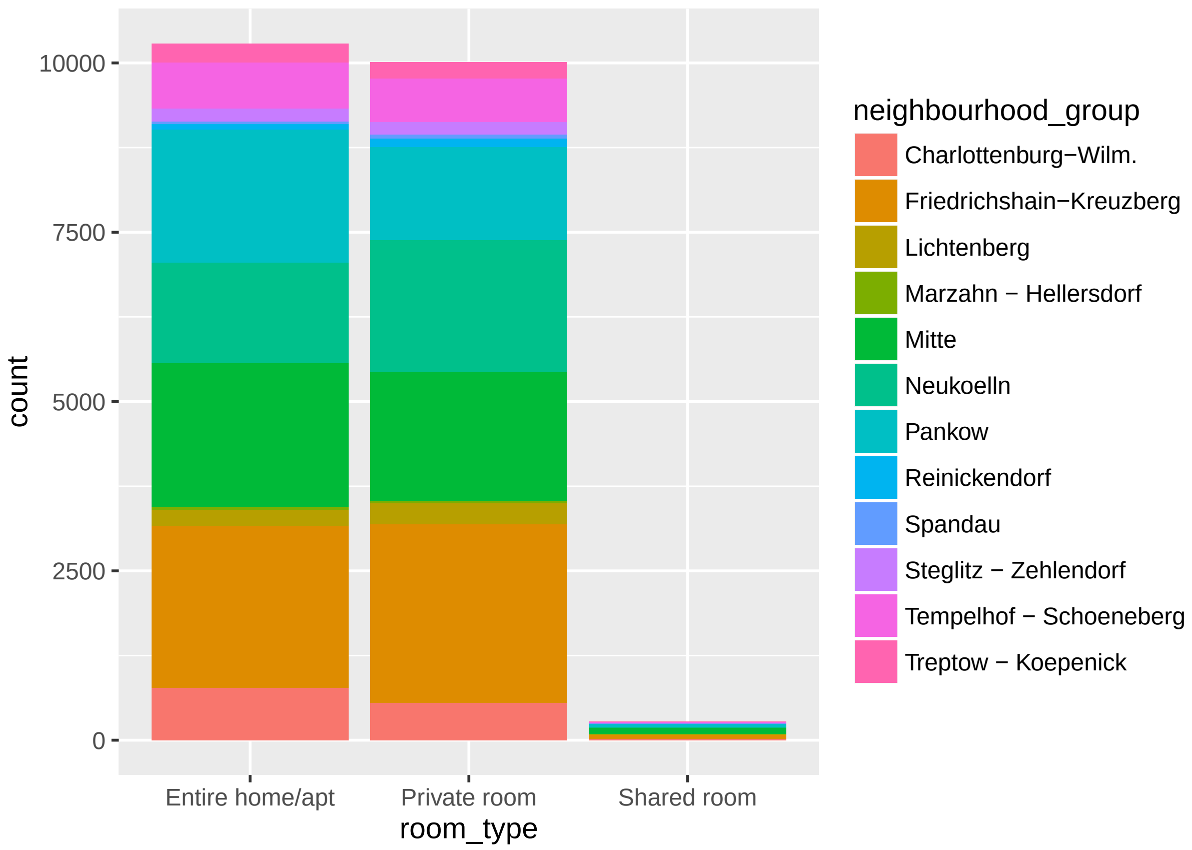
RLadies Berlin Hint: For the illustration of exact values in a bar plot use *stat = 'identity'* and add a definition of the y-value in the aesthetics *aes()*.

```
#a simple bar chart showing the distribution of the variable room type
ggplot(berlin, aes(room_type)) +
    geom_bar()
```



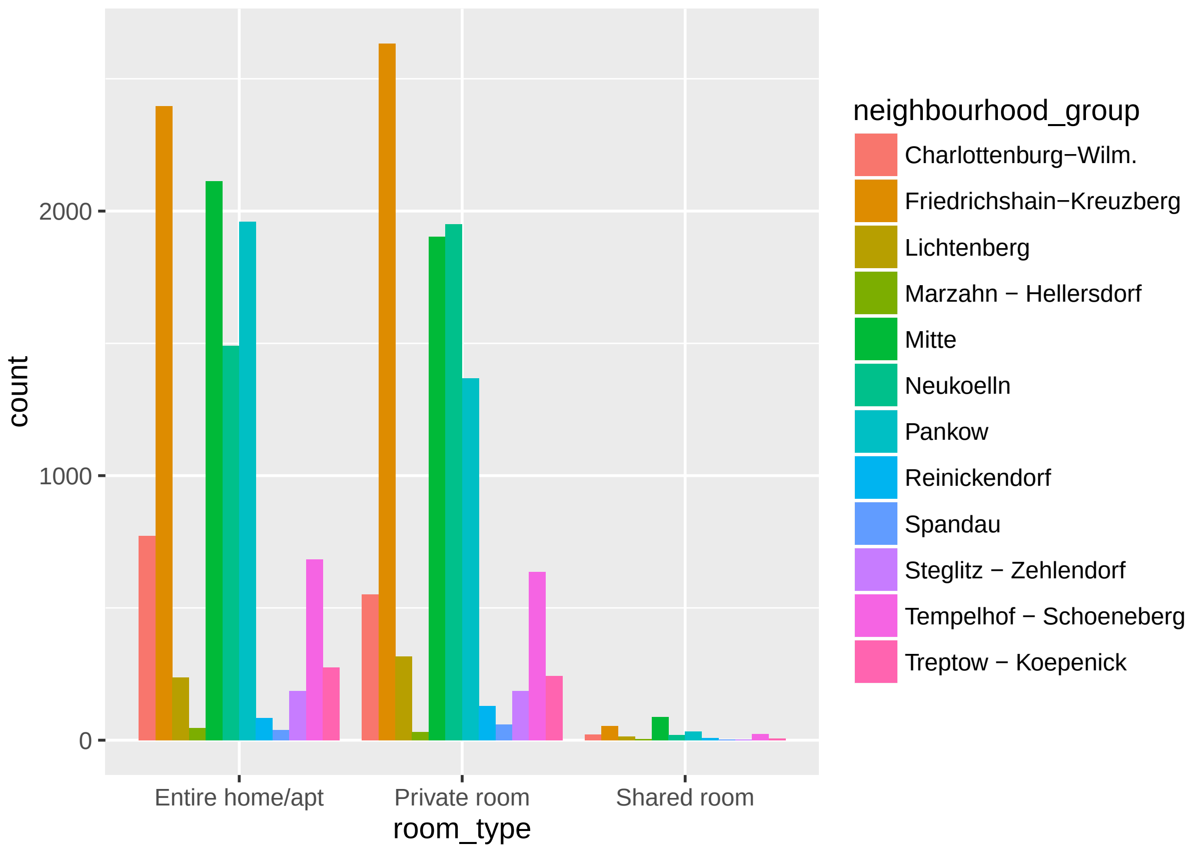
This is a bar chart showing the distribution of type of room splitted by neighbourhood. The parameter *fill* can be added either in the plot description or as aesthetics in the geom.

```
#distinguishing categories in bar charts
ggplot(berlin, aes(room_type, fill=neighbourhood_group)) +
    geom_bar()
```



Moving bars of one catogery next to each other by adding *position* parameter. Dodge means: Adjust position by dodging / moving overlaps to the side.

```r
#dodging bar charts next to each other
ggplot(berlin, aes(room_type, fill=neighbourhood_group)) +
    geom_bar(position="dodge")
```

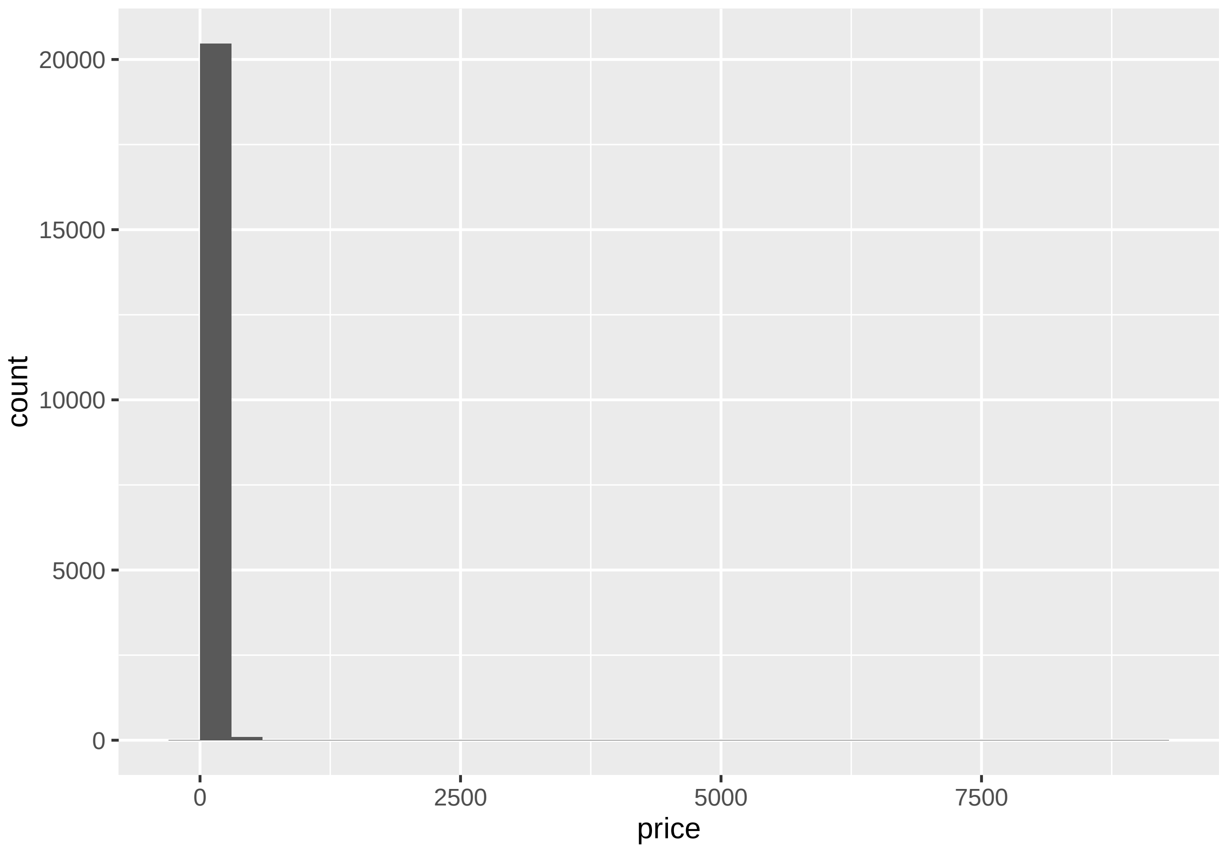


Histogram Another good tip from RLadies Berlin: Even though the default is to use bins that cover the range of the data, it is better to explore multiple widths to find the one that best suits your data.

Important parameters to keep in mind: binwidth, bin

```
#simple histogram showing the distribution of pricings for overnight stays in Berlin
ggplot(berlin, aes(price)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

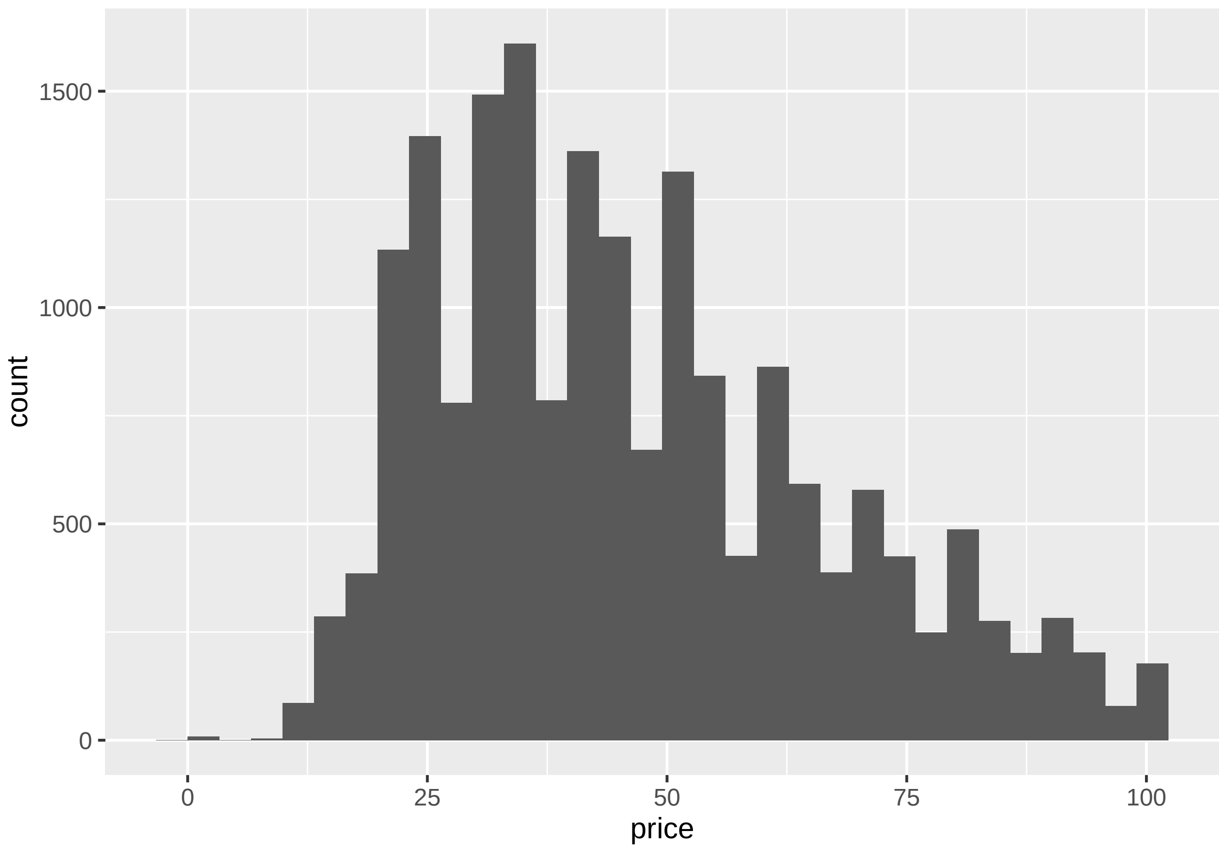


Even ggplot2 is suggesting us to change this parameter, as to better display the data.

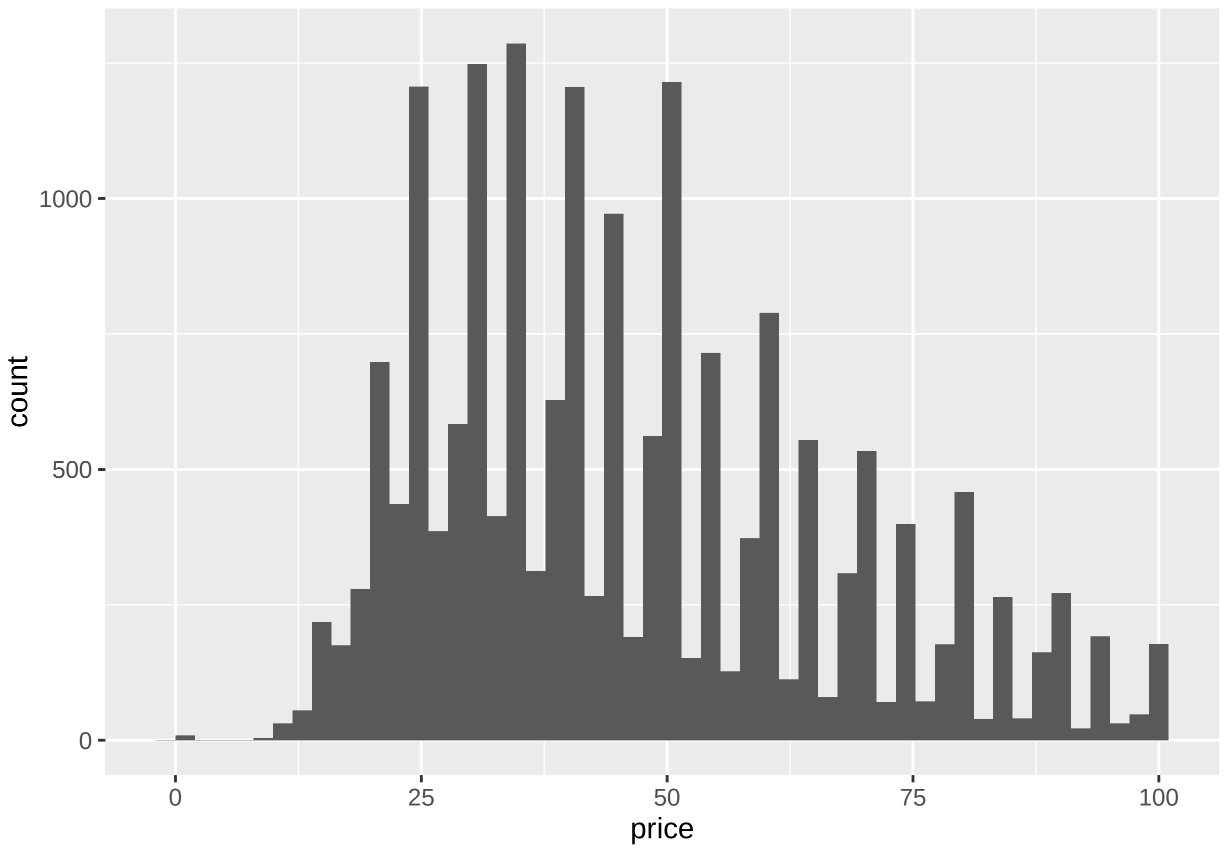
Let's do so, but let's focus on those homes whose price per night is < 100 EUR (just so the distribution looks nicer for visualization purposes).

```r
berlin_price <- subset(berlin,price<100)
ggplot(berlin_price, aes(price)) +
  geom_histogram()
```

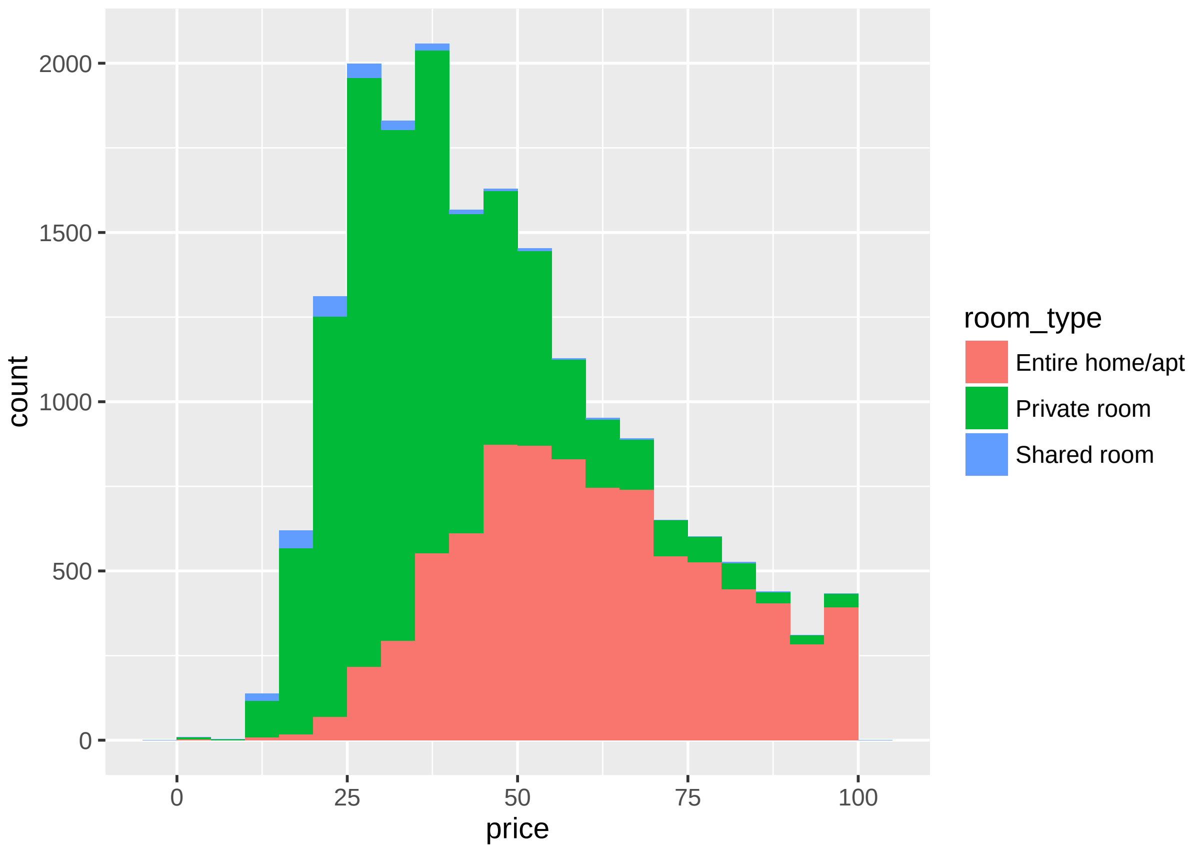
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
#definition of number of bins
ggplot(berlin_price, aes(price)) +
  geom_histogram(bins = 50)
```



```
#definition of binwidth
ggplot(berlin_price, aes(price, fill = room_type)) +
  geom_histogram(binwidth = 5)
```



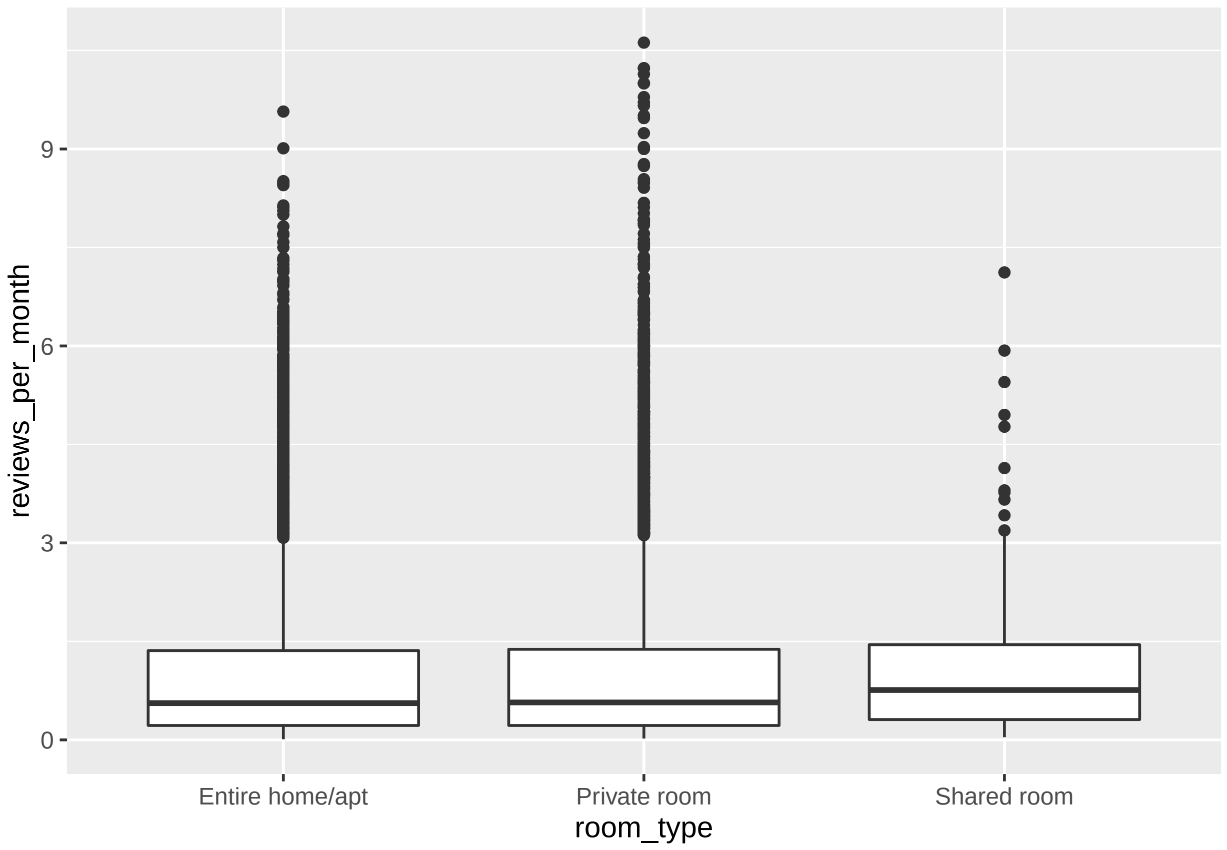
4 - Boxplot

Boxplots help display the distribution of variables across different groups, for example. The first and third quantiles values correspond to the actual limits of the box, while the median is depicted as the line inside the box.

This plot shows how many reviews per month each type of room gets.

```
ggplot(berlin, aes(x=room_type, y=reviews_per_month)) +
    geom_boxplot()
```

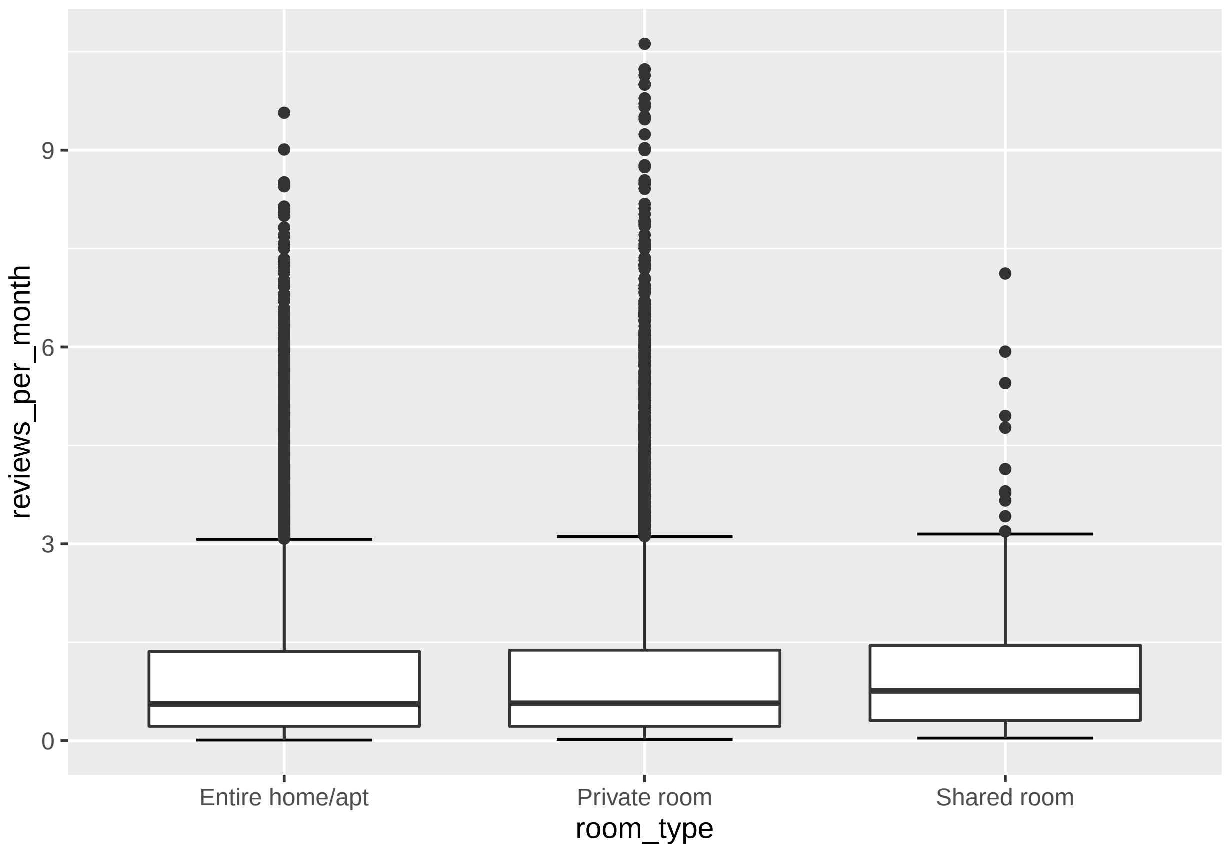
```
## Warning: Removed 4396 rows containing non-finite values (stat_boxplot).
```



In order to add a whisker, we need to use the integrated statistics **stat__xy** of **ggplot2**. Statistics calculate and return values based on the graph definition and are directly applied.

```
#stat_boxplot for defining whisker
ggplot(berlin, aes(x = room_type, y = reviews_per_month))  +
 stat_boxplot(geom ='errorbar', width = 0.65) +
 geom_boxplot()                 # or: stat_boxplot(geom='boxplot')
```

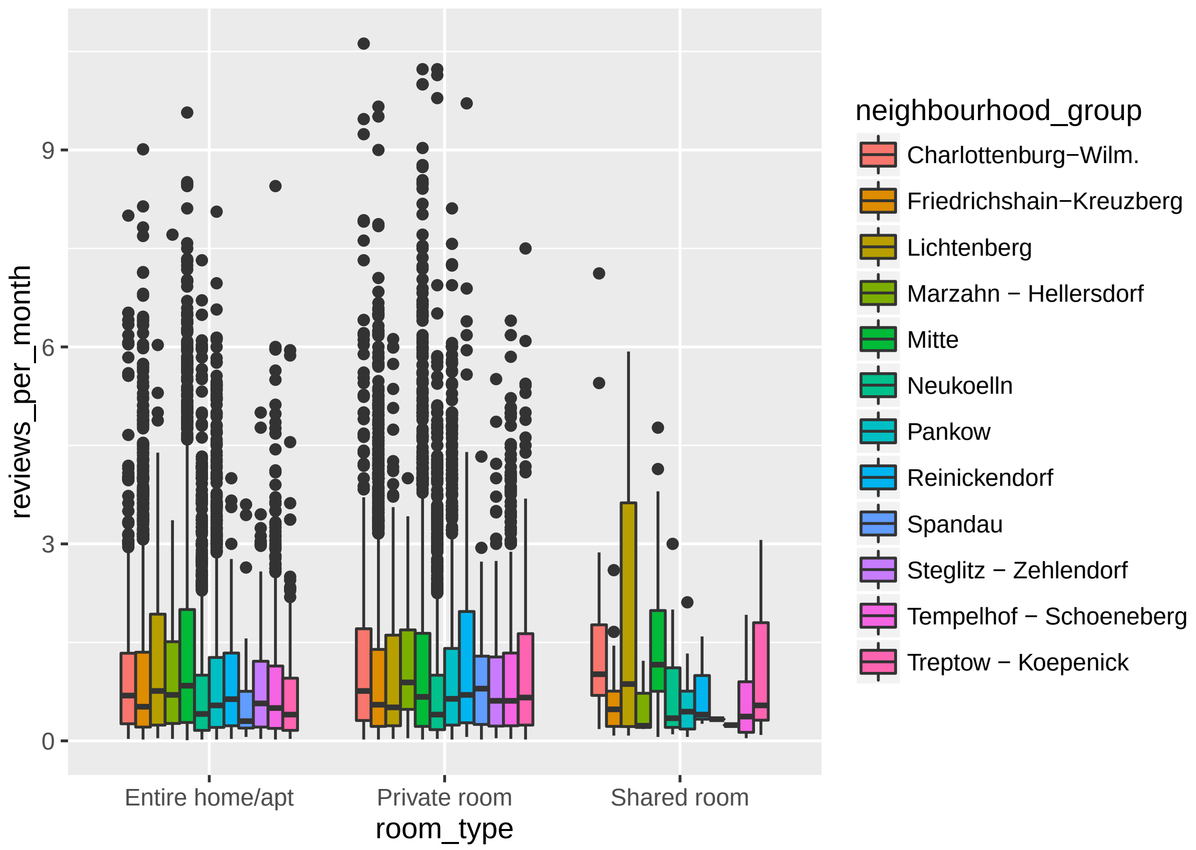
```
## Warning: Removed 4396 rows containing non-finite values (stat_boxplot).

## Warning: Removed 4396 rows containing non-finite values (stat_boxplot).
```



Also boxplots are distinguishable by another catogery using the *fill* in aesthetics *aes()*.

```
#boxplot grouping prices per color for each cut level, transformation y-axis into log-scale
ggplot(berlin, aes(x = room_type, y = reviews_per_month, fill = neighbourhood_group)) +
    geom_boxplot()
```

```
## Warning: Removed 4396 rows containing non-finite values (stat_boxplot).
```



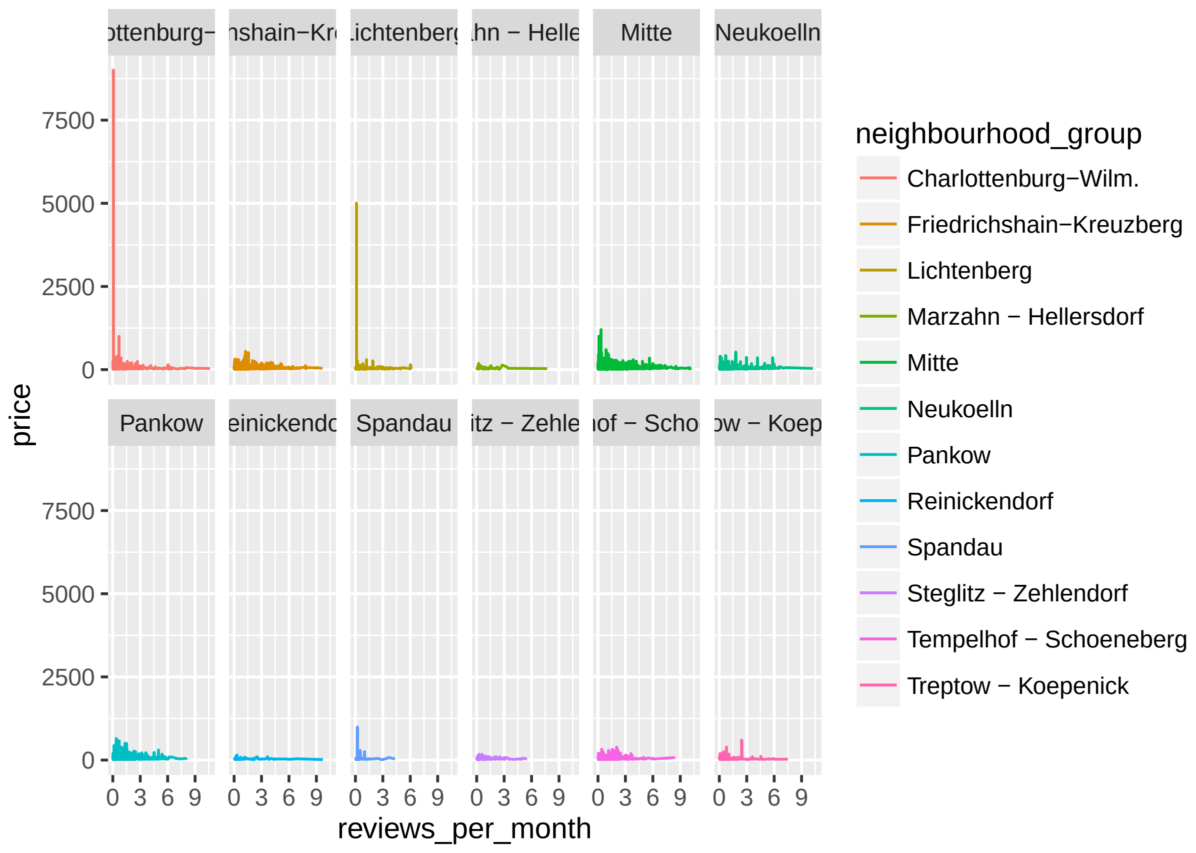
5 - Multiple facets

Finally, there are times when you need to split messy graphs by one or two factors in order to make better sense of them. This can be achieved in two different manners with **ggplot2**

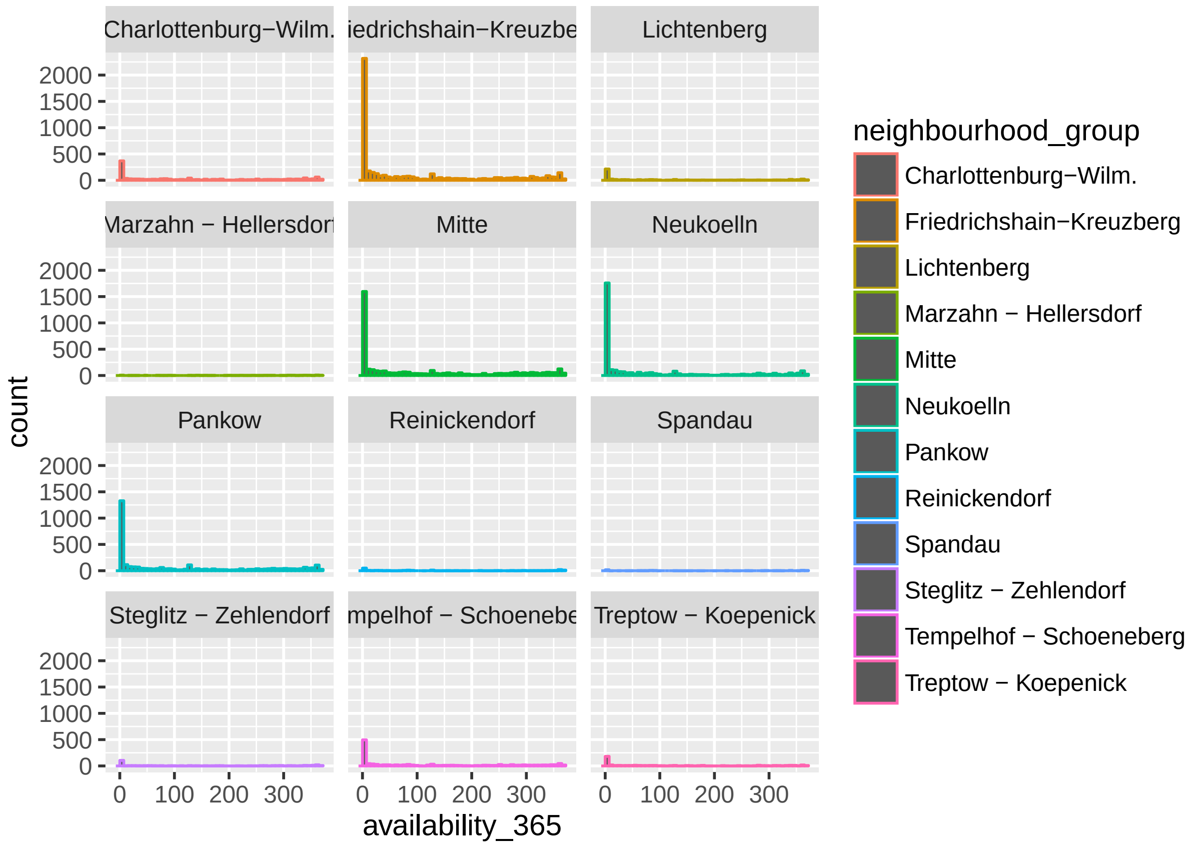
The first, **facet_wrap()**, arranges the separate facets subsequently next to each other. The number of rows and columns can be defined as additional parameter.

```r
#splitted by color and arranged in the two rows
ggplot(berlin, aes(x = reviews_per_month, y = price)) +
geom_line(aes(color = neighbourhood_group)) +
facet_wrap(~neighbourhood_group, nrow = 2)
```

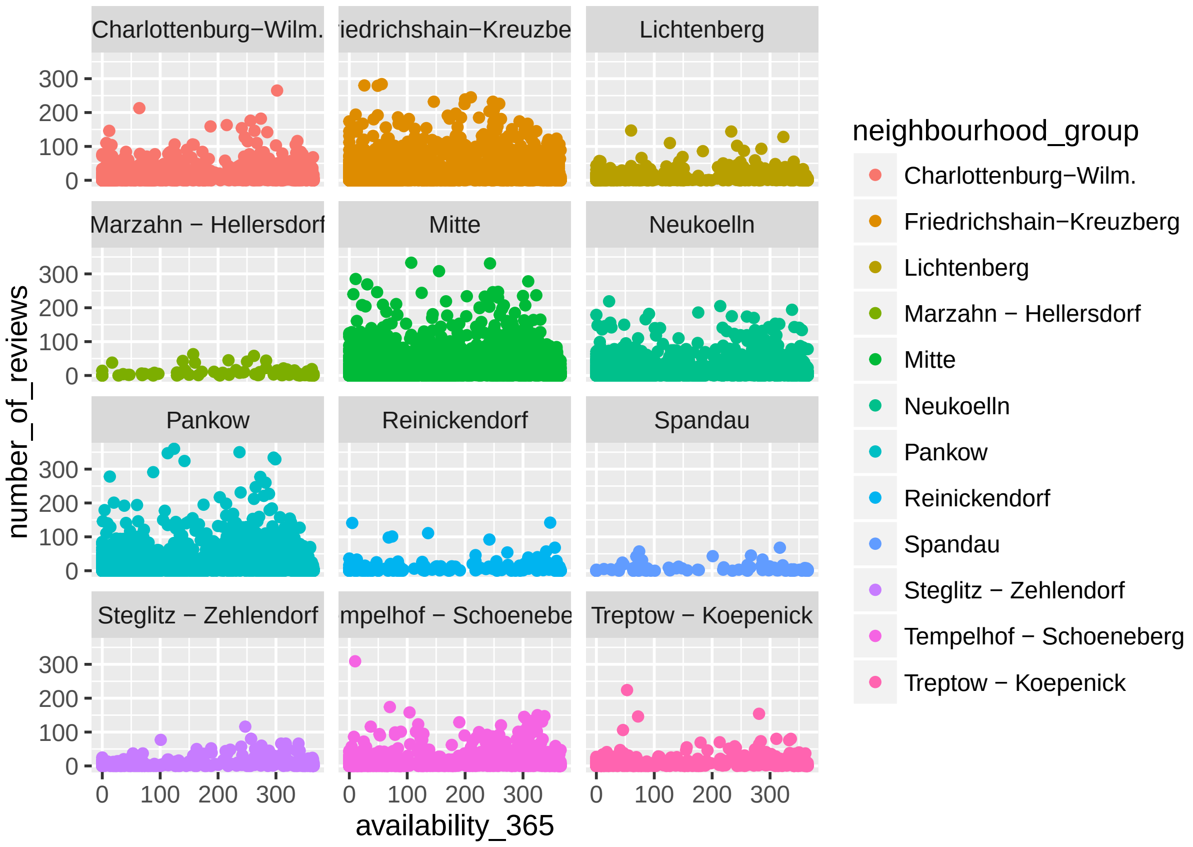
```
## Warning: Removed 4396 rows containing missing values (geom_path).
```



```r
ggplot(berlin, aes(x = availability_365)) +
geom_histogram(aes(color = neighbourhood_group),bins=50) +
facet_wrap(~neighbourhood_group, nrow = 4)
```



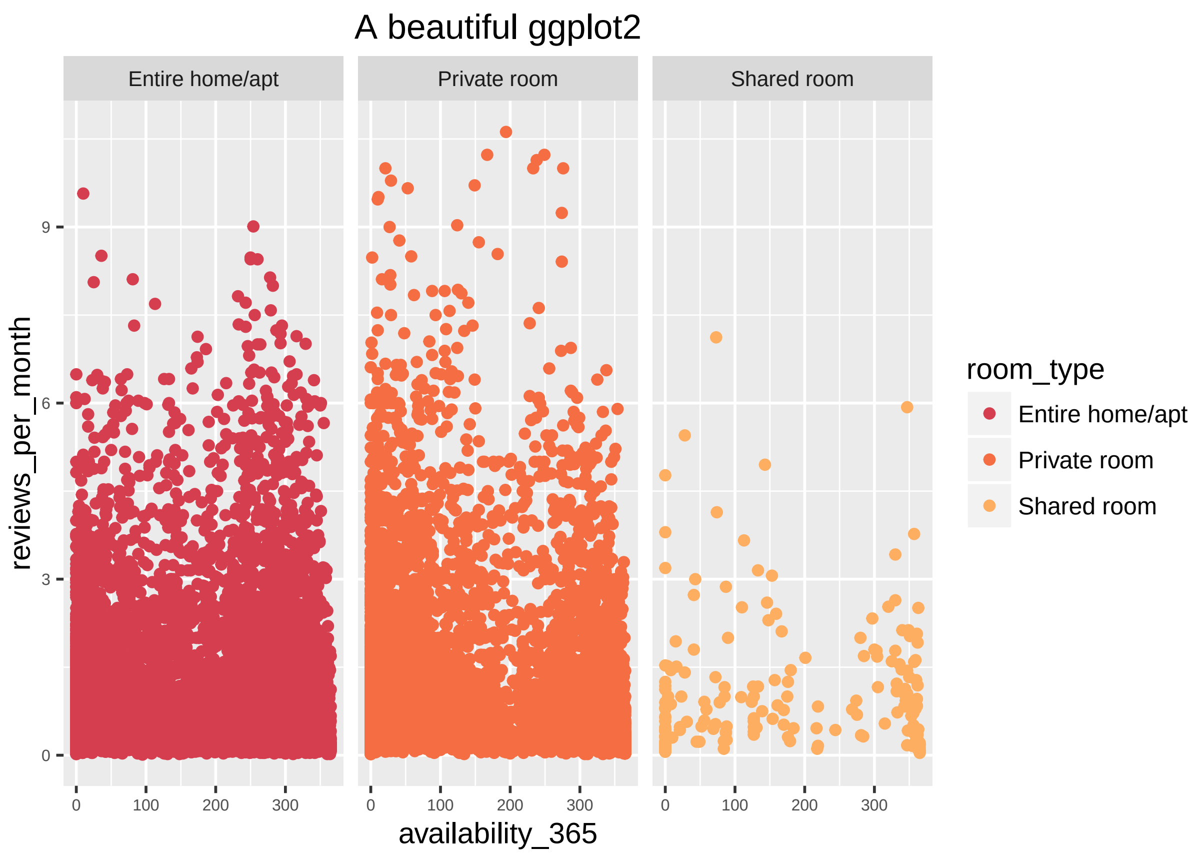
```r
ggplot(berlin, aes(x = availability_365, y = number_of_reviews)) +
geom_point(aes(color = neighbourhood_group)) +
facet_wrap(~neighbourhood_group, nrow = 4)
```



facet_grid() creates a matrix-based arrangement of our plots. Dataset factors are separated by the symbol "~". If there is only a single paramter applied as x coordinate, the second one has to be coded with a placeholder ".", e.g., + facet_grid(color ~ .).

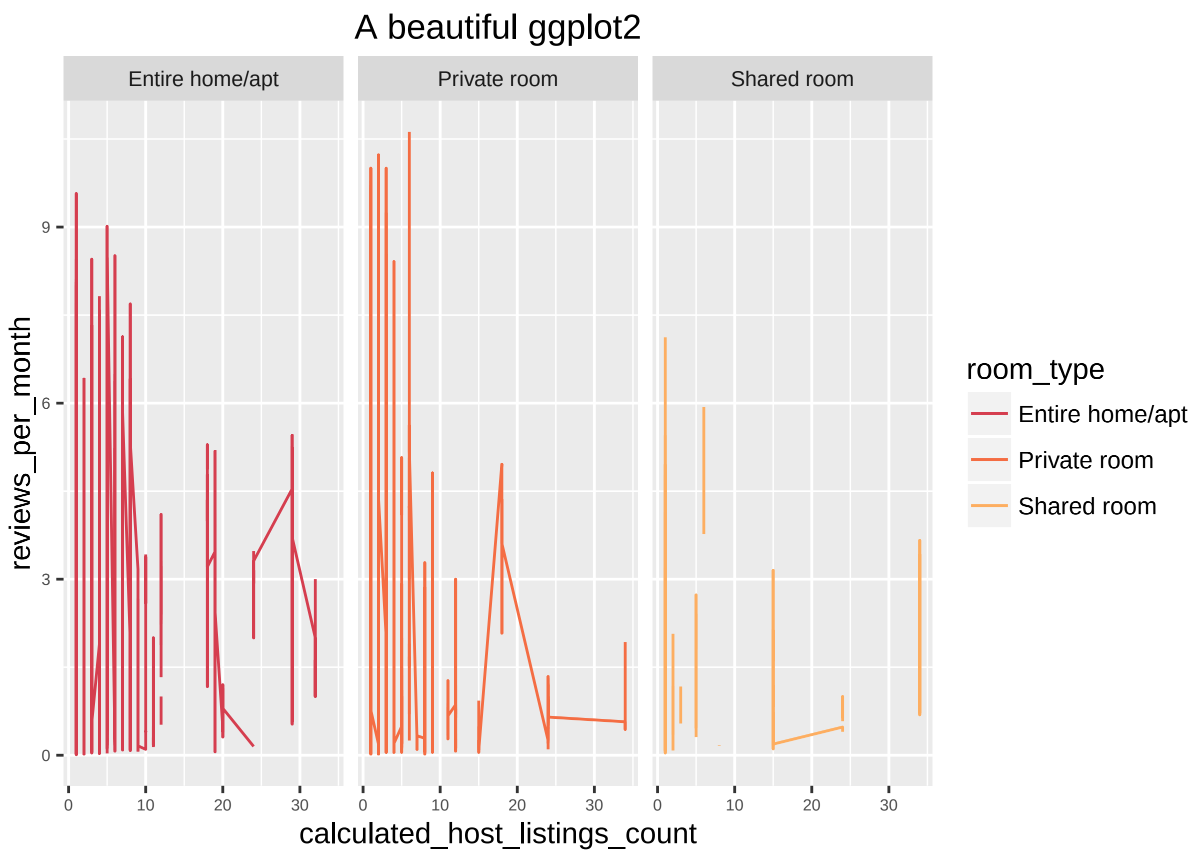
```
ggplot(berlin, aes(availability_365, reviews_per_month, color = room_type)) +
    geom_point() +
    facet_grid(.~room_type)+                           #splitted into grid / matrix (x ~ y)
    theme(axis.text = element_text(size = 6),          #modifying grid parameter: smaller text sizees
            strip.text = element_text(size = 8)) +
    scale_color_manual(values = brewer.pal(8, 'Spectral'))+       #changing colors using RColorBrewer
    ggtitle('A beautiful ggplot2')  #plot title
```

```
## Warning: Removed 4396 rows containing missing values (geom_point).
```



```r
ggplot(berlin, aes(calculated_host_listings_count, reviews_per_month, color = room_type)) +
    geom_line() +
    facet_grid(.~room_type)+                          #splitted into grid / matrix (x ~ y)
    theme(axis.text = element_text(size = 6),         #modifying grid parameter: smaller text sizees
            strip.text = element_text(size = 8)) +
    scale_color_manual(values = brewer.pal(8, 'Spectral'))+      #changing colors using RColorBrewer
    ggtitle('A beautiful ggplot2')  #plot title
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



Useful commands

Here are more common commands and parameters of **ggplot2**.

```
# coord_flip()                                              - rotates your plot
# theme_bw()                                                    - a black and white scheme
# aes(shape = , color = )                               - define a variable to this parameter inside aes() 
# aes(fill = )                                              - use to distinguish a variable in a bar pl
# scale_fill/color_manual()                         - define your own colors
```

Practice time

Now it’s time for you to play a little with our dataset and the little we covered today :)

Here are 4 small tasks for you to complete:

Exercise 1 Description of the plot: *data: full dataset* x-axis: room type *y-axis: availability within the next 365 days

```
#your code would be here :)
```

Exercise 2 Description of the plot: *data: only those BnB homes cheaper than 100EUR per night* plot: histogram *x-axis: price* color and facetting by: type of room, 3 rows

```
#your code would be here :)
```

Exercise 3 Description of the plot: *data: full dataset* plot: boxplot *x-axis: type of room* y-axis: reviews per month *color and grouping by: type of room* facetting by: neighbourhood group, 4 rows

```
#your code would be here :)
```

Exercise 4 Description of the plot: *data: only those BnB homes located in the city center that might only be rented for longer than 20 days* plot: scatter plot *x-axis: price* y-axis: number of reviews *facetting: type of room, using facet_grid* plot title: Price and number of reviews from listings in Berlin's city center

```
#your code would be here :)
```

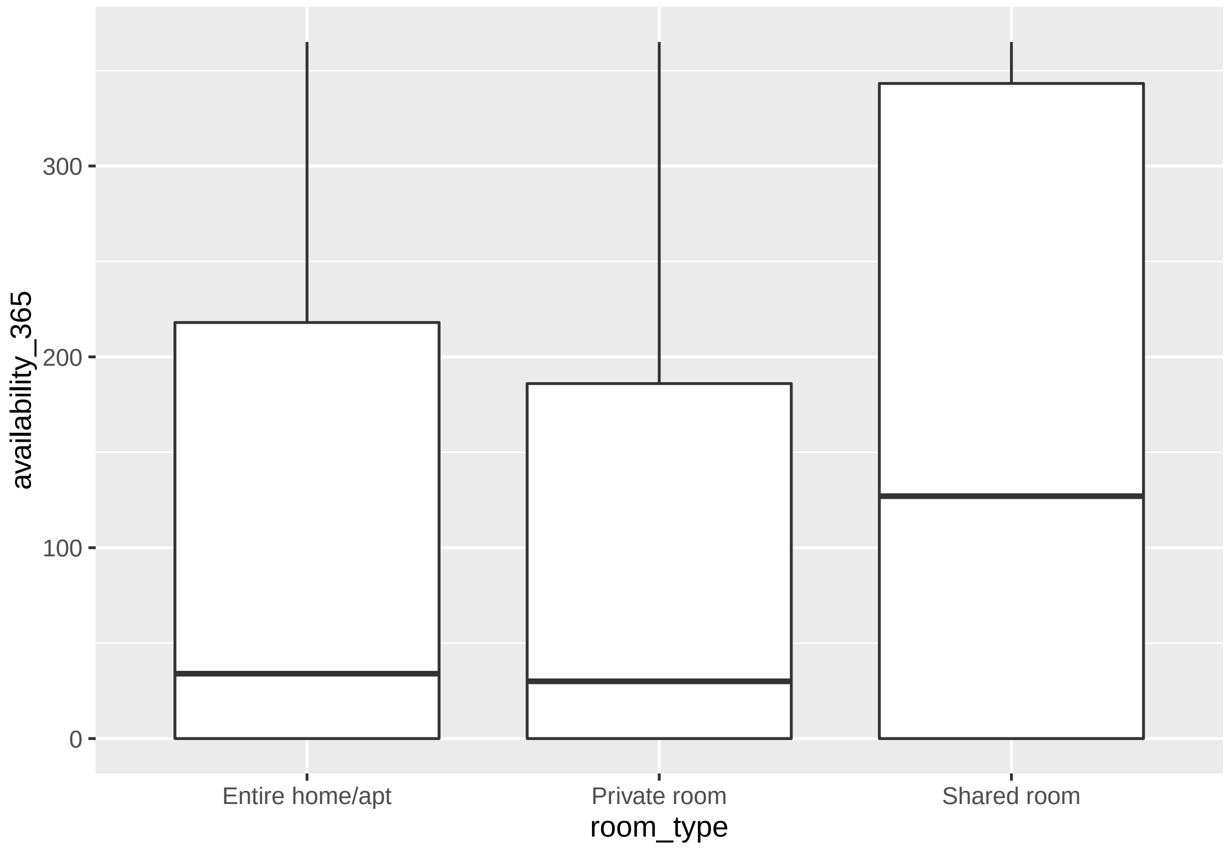
More examples, help and used resources

Here are some further links covering stuff we had no time to go over, and also some we used while preparing this workshop. Sharing is caring!

- Cookbook for R
- ggplot2 cheat sheet
- Another amazing cheat sheet
- stackoverflow
- ggplot2 Harvard tutorial
- RLadies Berlin tutorial: Hands-on data visualization with ggplot2
- *Bonus*: Feel like adding dogs to your ggplots? Check this out! Your plots will never feel alone anymore #heartemoji ***

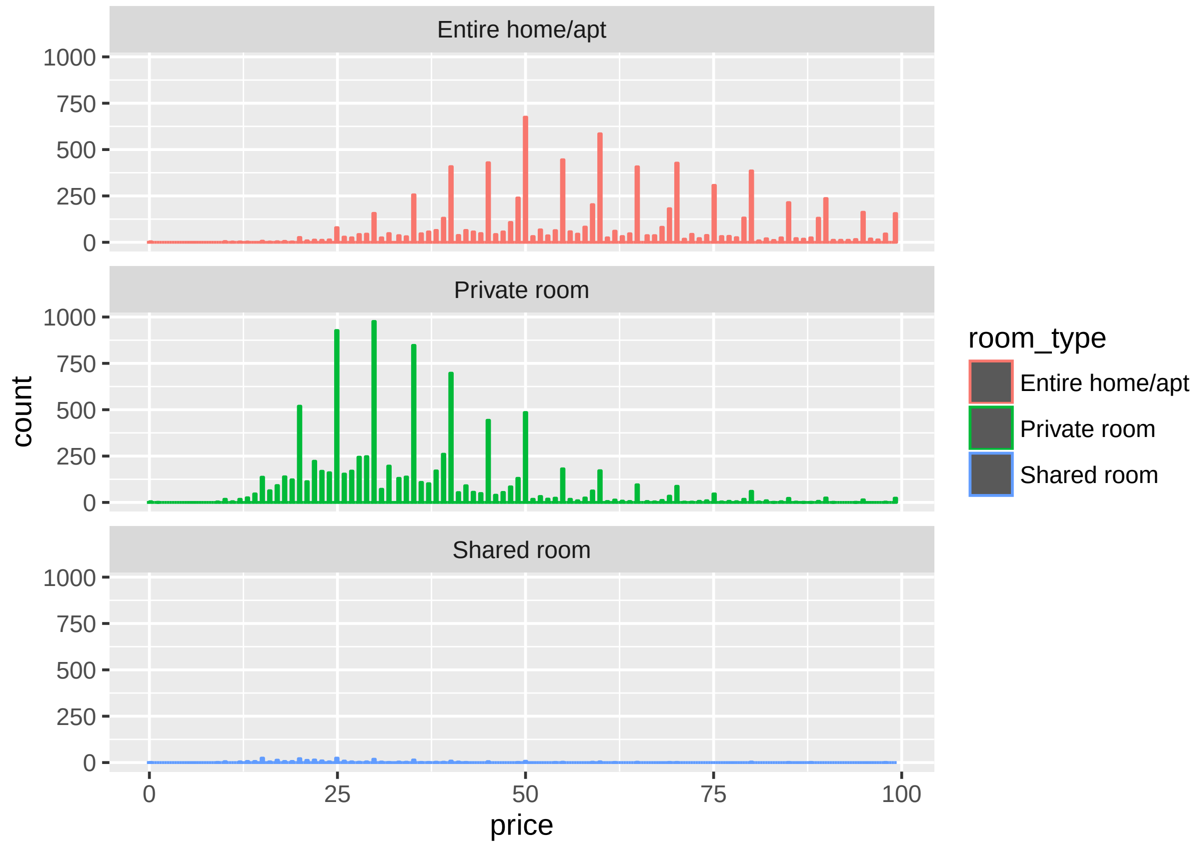
Solutions to practice exercises

```
ggplot(berlin, aes(x = room_type, y = availability_365))  +
 geom_boxplot()
```



Practice 1

```r
berlin_price <- subset(berlin, price < 100)
ggplot(berlin_price, aes(x = price)) +
geom_histogram(aes(color = room_type),bins=300) +
facet_wrap(~room_type, nrow = 3)
```

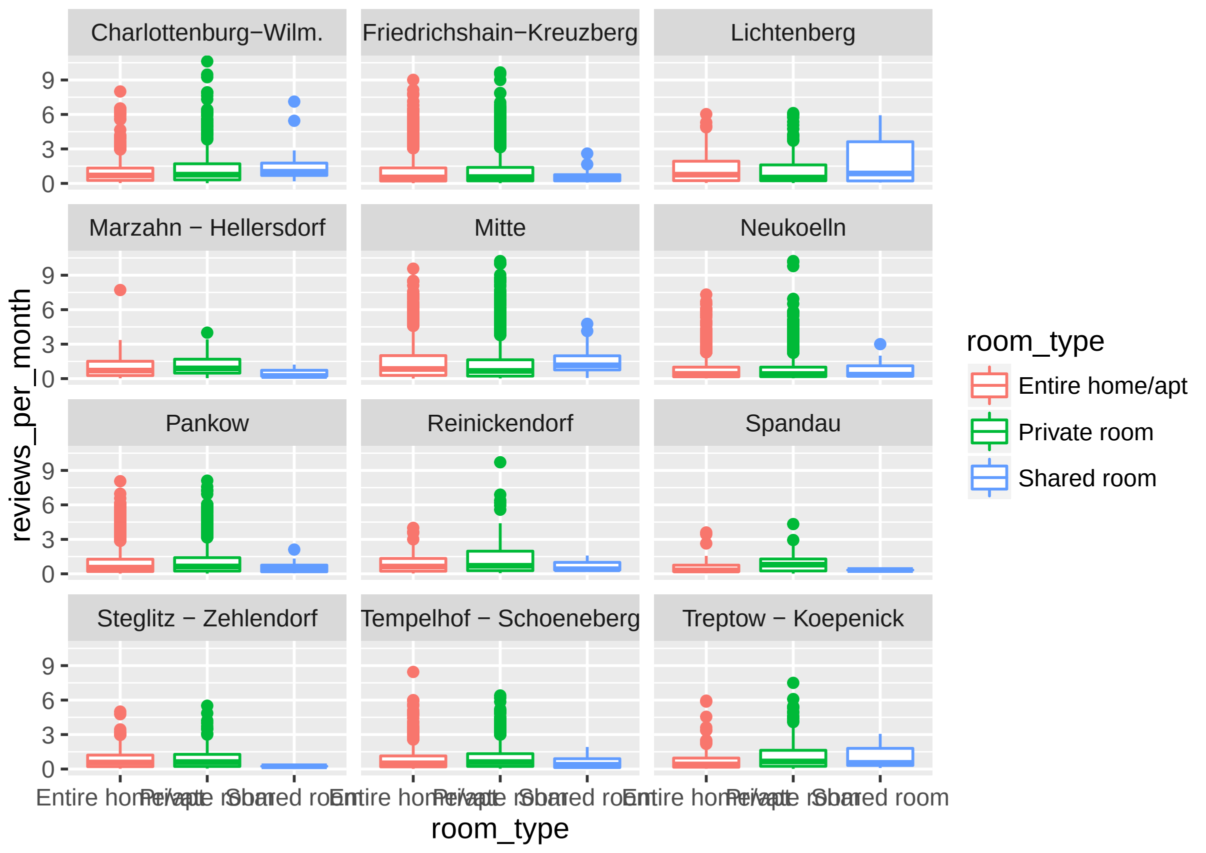


Practice 2

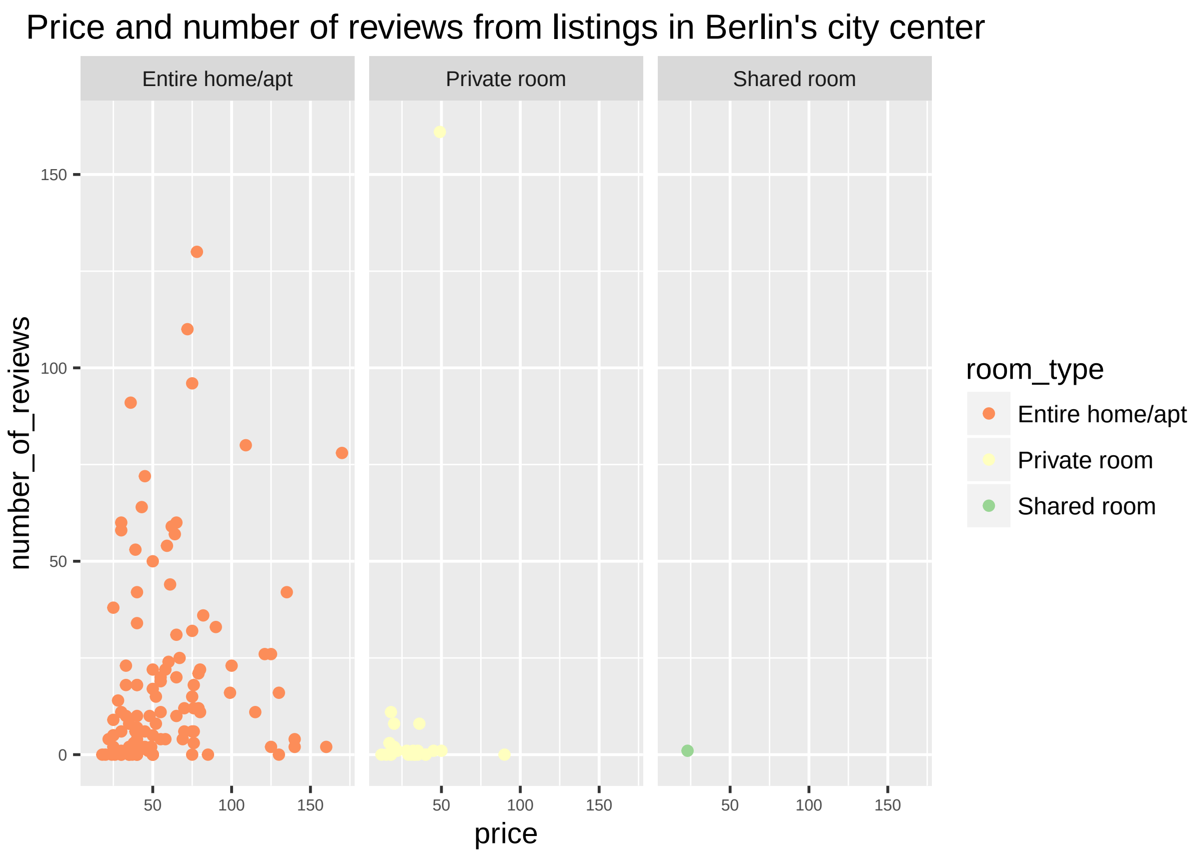
```r
ggplot(berlin, aes(x = room_type, y = reviews_per_month)) +
    geom_boxplot(aes(color = room_type)) +
  facet_wrap(~neighbourhood_group, nrow = 4)
```

Practice 3

```
## Warning: Removed 4396 rows containing non-finite values (stat_boxplot).
```



```
berlin_neighborhood <- subset(berlin,neighbourhood_group=="Mitte"&minimum_nights>20)
ggplot(berlin_neighborhood, aes(price, number_of_reviews, color = room_type)) +
    geom_point() +
    facet_grid(.~room_type)+                          #splitted into grid / matrix (x ~ y)
    theme(axis.text = element_text(size = 6),         #modifying grid parameter: smaller text sizees
            strip.text = element_text(size = 8)) +
    scale_color_manual(values = brewer.pal(3, 'Spectral'))+      #changing colors using RColorBrewer
    ggtitle('Price and number of reviews from listings in Berlin\'s city center')    #plot title
```


Price and number of reviews from listings in Berlin's city center
Entire home/apt
Private room
Shared room
number_of_reviews
0
50
100
150
price
50
100
150
room_type
Entire home/apt
Private room
Shared room

Practice 4