Lab 1

STAT 131A

January 24, 2022

Welcome to the first lab of STAT 131A!

The goal of the computational part of the labs is to get you familiar with the data analysis you have already seen in lectures and produce them in R by yourself. Most of the exercises in the lab will be a simplified version of what is done in the lectures.

You may want to go ahead and compile this lab (the "knit" buttom). That will give you a pdf version to look at if you want (and verifies you aren't having any problem compiling the file before you make changes).

But generally you should go through the .Rmd and go through the code and make sure you understand it, then do the exercise. Interactively, you will probably just hit run for each chunk. But once you have successfully completed the exercise (i.e. ready to turn it in), reknit the document (in pdf) to make sure you did everything right. That is much better than doing the whole thing, and only at the end (when you want to turn it in!) discovering you have to go back because there was an error somewhere from earlier.

Submitting the lab

Instructions for code: We have already put in R chunks for where your code should go. For some questions, we will ask you to save your answer as a particular variable. For example, we might give you a code chunk that looks like this:

```r
set.seed(4291)
# insert code here save the median of your simulated data as
# 'medx'
```

And you might complete it like so:

```r
set.seed(4291)
# insert code here save the median of your simulated data as
# 'medx'
x <- rnorm(1000)
medx <- median(x)
medx
```

```
## [1] 0.01612433
```

Misspelling of the requested variable name may result in errors when answers are pulled for subsequent parts or for scoring purposes. In general objects that are not large datasets which take up a page or more should be called at the end of the code block in which they are asked for, for scoring purposes. Thus, it is a good idea to put the variable name at the bottom so it prints (assuming its not a huge object), and usually this should be already part of the provided code. It also helps you check your work.

Of note: Sometimes an exercise will ask for code AND posit a question. Make sure that if the answer to the question is not an output of the code, then you must answer it separately outside of the code block. For example the problem might ask you to make a plot and describe its prominent features. You would write the code to make the plot, but also write a sentence or two outside of the code block (plain text) to describe the features of the plot.

Submission: We are asking you to submit both the *PDF* and *.Rmd* to gradescope. **Do not modify your file name**. Otherwise, gradescope would fail to recognize it. For example, you will submit `LAB01.pdf` and `LAB01.Rmd` for this lab.

Exploratory Analysis – Rent Price in SF Bay Area

According to a U.S. News and World Report in 2016 in collaboration with Zillow, the off-campus housing price of Stanford and Berkeley have ranked No.1 and No.3 among top-ranked universities. Student renters in the Bay area expect to see the highest annual rental appreciation, which is anticipated to grow by more than 6 percent over the coming year. (Reference: College Students Can Expect Highest Off-Campus Housing Costs in Palo Alto and Princeton, Lowest in St. Louis) The rent price of Berkeley and its surrounding cities may vary. Many students choose to live in the neighborhoods such as Albany, Oakland, El Cerrito and Richmond. In this lab, we will explore a dataset scraped from Craigslist posting with the rent prices in Stanford and Berkeley, as well as their nearby cities. The dataset is a subset which only includes apartment/housings with less than or equal to 4 bedrooms.

The table **craigslist.csv** contains a Simple Random Sample (SRS) of the monthly rent price in 2016. Each posting record contains the following information:

- time: posting time
- price: apartment/housing monthly rent price
- size: apartment/housing size (ftˆ2)
- brs: number of bedrooms
- title: posting title
- link: posting link, add "https://sfbay.craigslist.org" to visit the posting page
- location: cities

Read in data.

```
craigslist <- read.csv("craigslist.csv",
                       header = TRUE, stringsAsFactors = FALSE)
#Check to see if everything looks as expected:
head(craigslist)
```

```
##                  time price size brs
## 1 2016-09-27 18:54:00  4100   NA   2
## 2 2016-09-27 18:53:00  1090  400   1
## 3 2016-09-27 17:29:00  3275  706   1
## 4 2016-09-27 17:29:00  4150  958   2
## 5 2016-09-27 17:25:00  2491  735   1
## 6 2016-09-27 17:25:00  2825  716   1
##                                                      title
## 1    Furnished Townhouse in the Elmwood  Available March 8 2017
## 2                       Private/small studio ALL UTILITIES INCLUDED
## 3     Luxury  Convenience all at Parker Movein special Hurry in
## 4 New luxury 2BR in Downtown Berkeley  1500 off 1st months rent
## 5                  One Bedroom Ready For You 500 off Move In Cost
## 6                                     Ready Now for Immediate Movein
##                        link location
## 1 /eby/apa/5802541604.html berkeley
## 2 /eby/apa/5802540177.html berkeley
## 3 /eby/apa/5802443907.html berkeley
## 4 /eby/apa/5802443556.html berkeley
## 5 /eby/apa/5802429182.html berkeley
## 6 /eby/apa/5802414465.html berkeley
```

Cities included in the dataset.

```
unique(craigslist$location)
```

```
##  [1] "berkeley"            "oakland"             "albany / el cerrito"
##  [4] "richmond"            "emeryville"          "alameda"
##  [7] "palo alto"           "mountain view"       "sunnyvale"
## [10] "menlo park"          "redwood city"
```

A posting example (i.e. single observation).

```
craigslist[1, ]
```

```
##                   time price size brs
## 1 2016-09-27 18:54:00  4100   NA   2
##                                                         title
## 1 Furnished Townhouse in the Elmwood  Available March 8 2017
##                        link location
## 1 /eby/apa/5802541604.html berkeley
```

Summary statistics

Exercise 1.

Computing summary statistics is always the first step in the exploratory analysis. The summaries may include average, median, maximum, minimum, etc. One simple method is to use the `summary` function.

(a) Find out the number of postings in the dataset.

```
# insert code here
```

(b) Use `summary` function to get the mean, median, maximum and minimum of the monthly rent.

```
# Insert your code here:
```

(c) Use `table` function to get the number of postings in each city.

```
# Insert your code here:
```

(d) What percentage of entries were more than $3,000 per month?

```
# Insert code here:
```

Histograms and boxplots

Recall that many basic R plot functions accept the same options, such as:

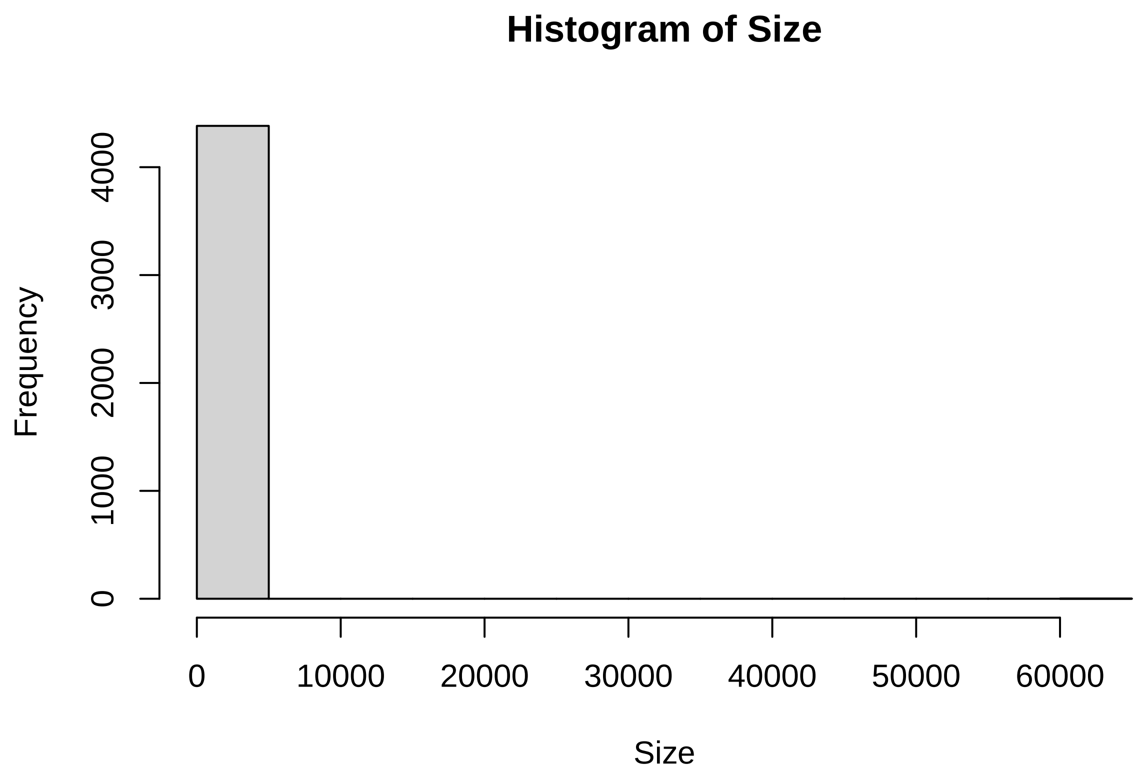
- `main`: Title of the plot.
- `xlab/ylab`: x/y axis label.
- `col`: plot color, the usuage of this variable may vary for different plot functions. Here the two color corresponds to female and male.
- `xlim/ylim`: the limits (starting and ending values) for the x/y axis.
- `legend`: the legend of the plot, the usuage of this variable may vary for different plot functions.

Always remember to add the title and axis label when you create a plot. You can check the help document (e.g `?hist`) to experience other arguments.

Histograms

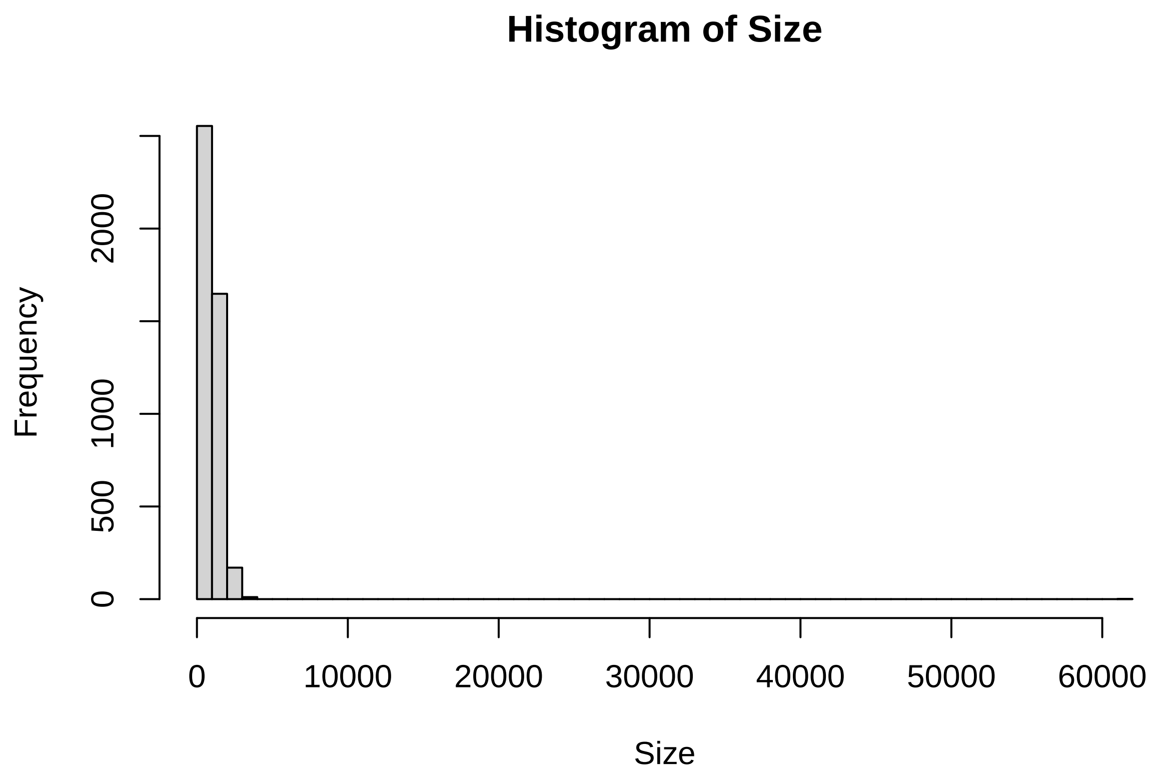
For example, to adjust plot of the histogram of our simulated `sample` in the previous probability section:

```
hist(craigslist$size,
     main = "Histogram of Size", # plot title
     xlab = "Size")
```



You can see that this is not very informative with just one big peak and then not much more. We can change the number of breaks:

```
hist(craigslist$size,
     main = "Histogram of Size", # plot title
     xlab = "Size", breaks=50)
```



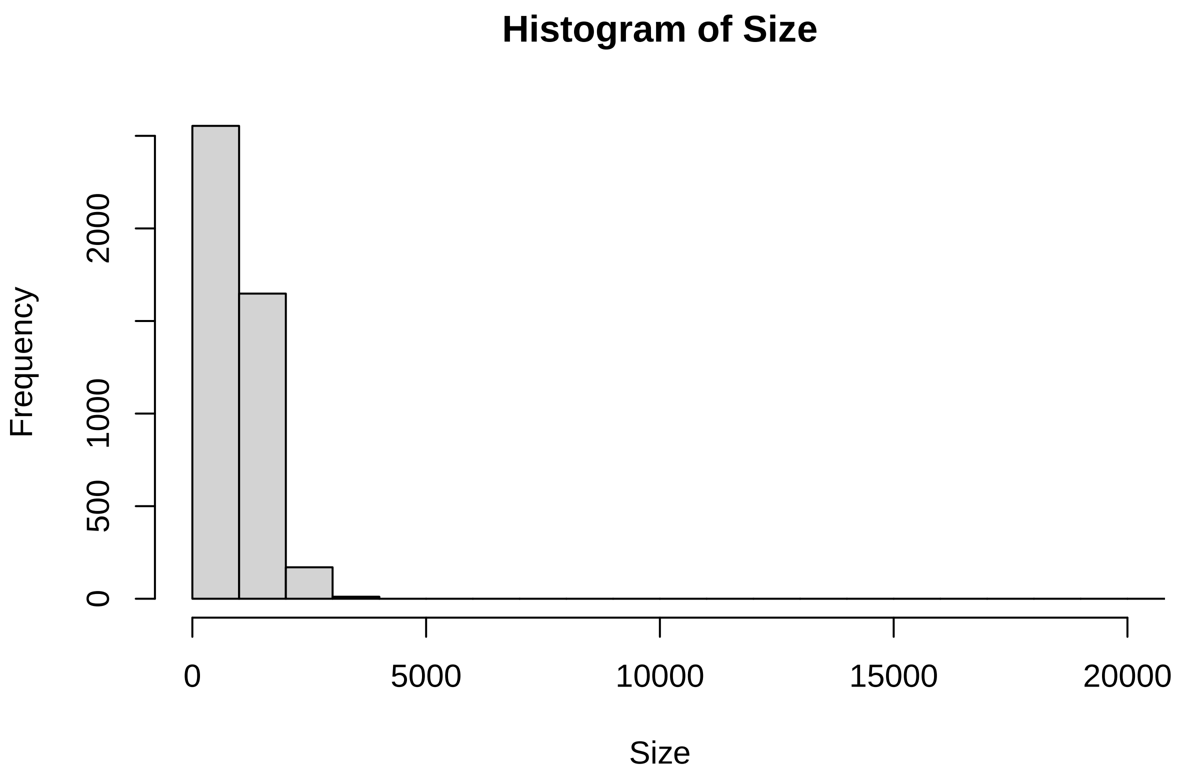
We see that most of the data is much smaller. Indeed if we do a summary, we see this:

```
summary(craigslist$size)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##       1     735     947    1053    1200   62000    1492
```

We can choose to do a "zoom" on the section with the data:

```
hist(craigslist$size,
     main = "Histogram of Size", # plot title
     xlab = "Size", breaks=50,
     xlim=c(0,20000))
```

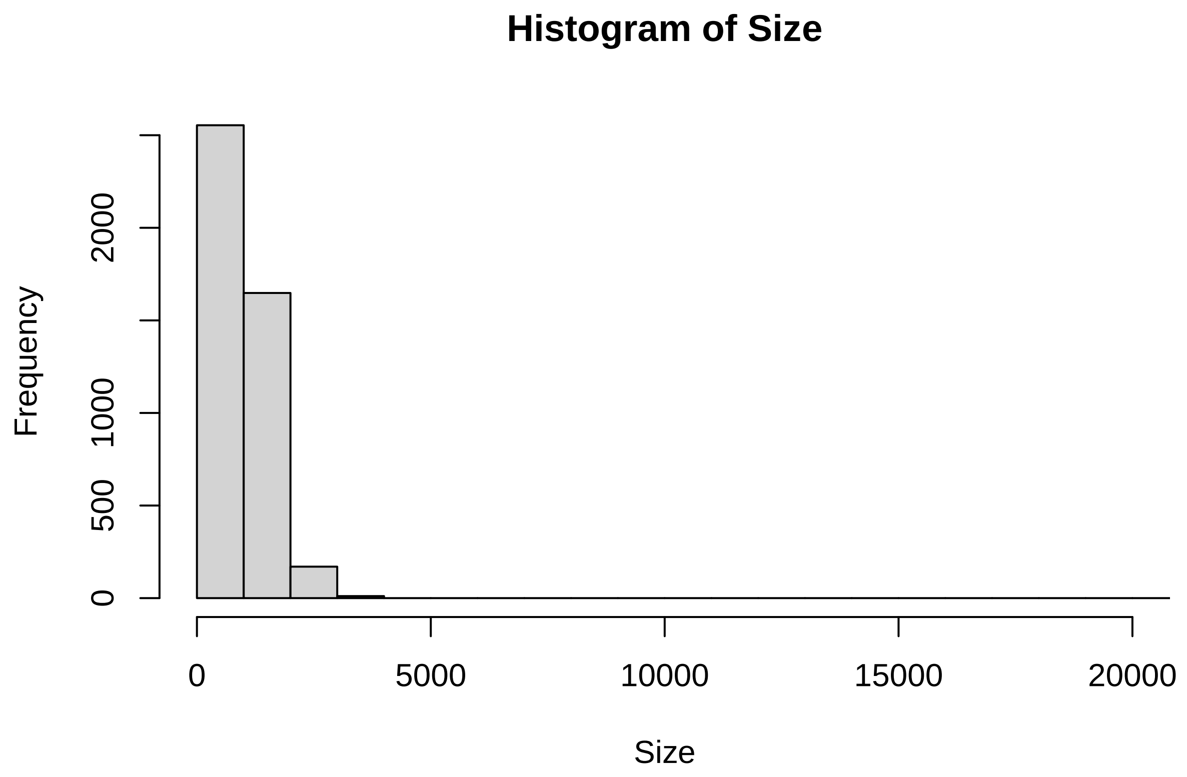


Notice that the `breaks` argument is based on the *whole range of the data*, and you are just zooming. So when you zoom, you need to often increase the breaks.

Exercise 2.

Fix the code above so that you have more reasonable number of breaks and zoom up further into the data. There is no singular right answer, just make it more or less reasonable than what it currently is.

```r
# Fix this code:
hist(craigslist$size,
     main = "Histogram of Size", # plot title
     xlab = "Size", breaks=50,
     xlim=c(0,20000))
```



Exercise 3.

(a) Plot a histogram of the monthly rent. Choose a number of breaks you think best from 10, 50, 300.

```
#Insert code here:
```

(b) Plot another histogram with the argument `freq = FALSE`. What is the interpretation of y axis numbers? Hint, be careful, there are two common answers, one of which is wrong, but can be easily confused with the right answer.

Hint: It is helpful to check the help document using `?` if you are confused about one argument or usage. Always remember to add the title and axis label when you create a plot.

```
#Insert code here:
```

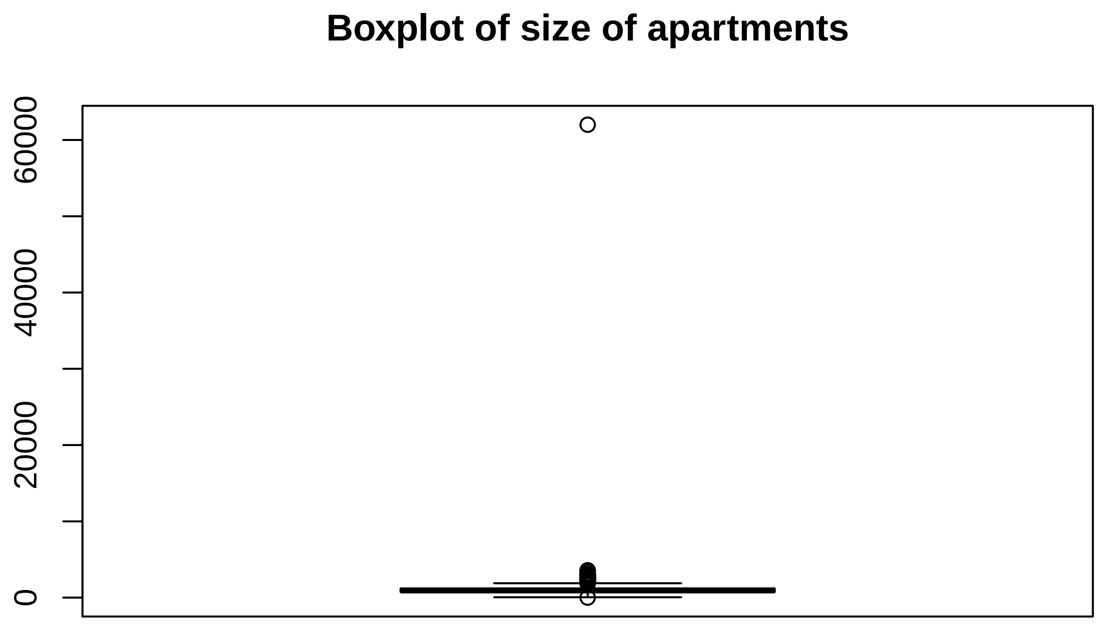
Your non-coding answer:

```
*Add text here*
```

Boxplots

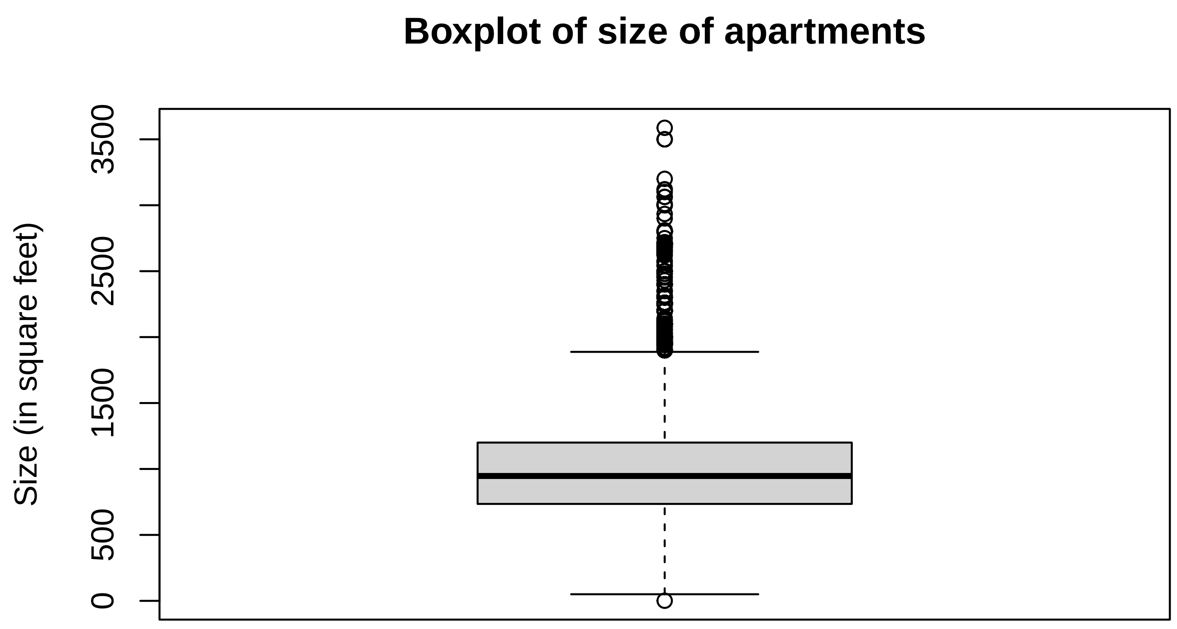
For a boxplot, we use the `boxplot` function.

```
boxplot(craigslist$size,
     main = "Boxplot of size of apartments"
)
```



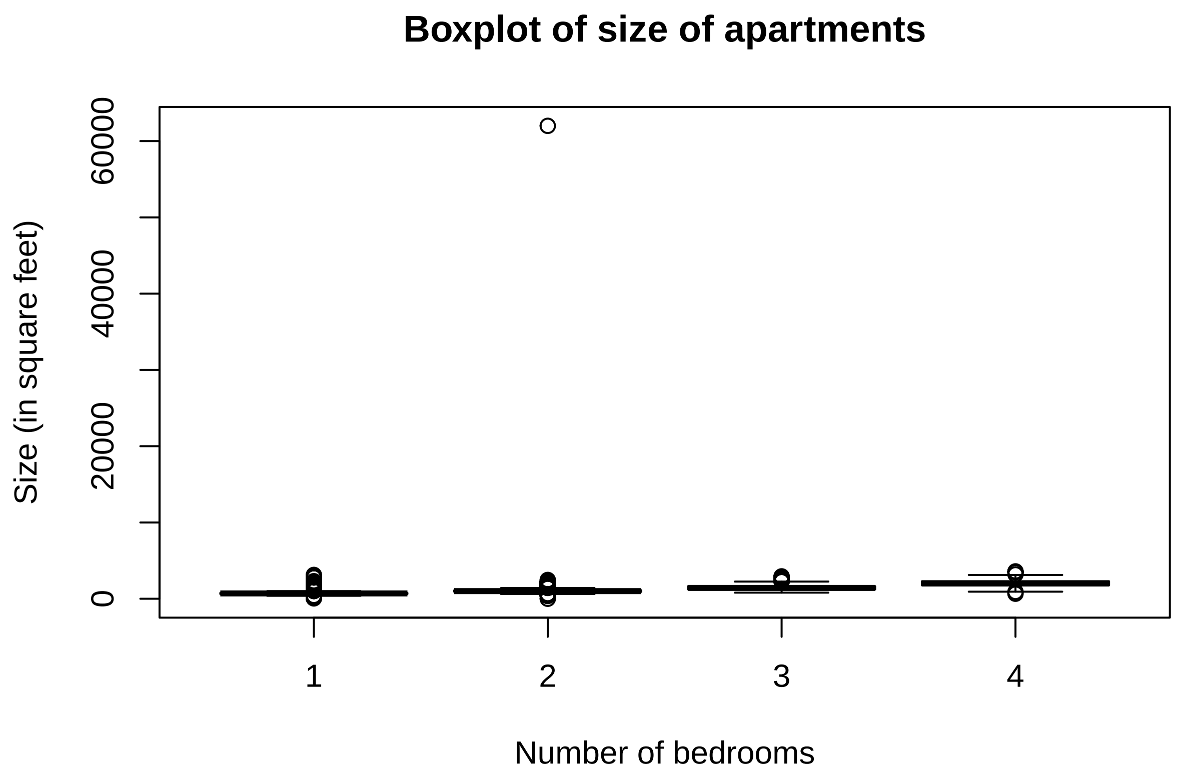
Notice how we can really see that one observation which was messing up our histogram, and now our boxplot too. Let's just take that observation out

```
boxplot(craigslist$size[craigslist$size<60000],
     main = "Boxplot of size of apartments",
     ylab="Size (in square feet)"
)
```



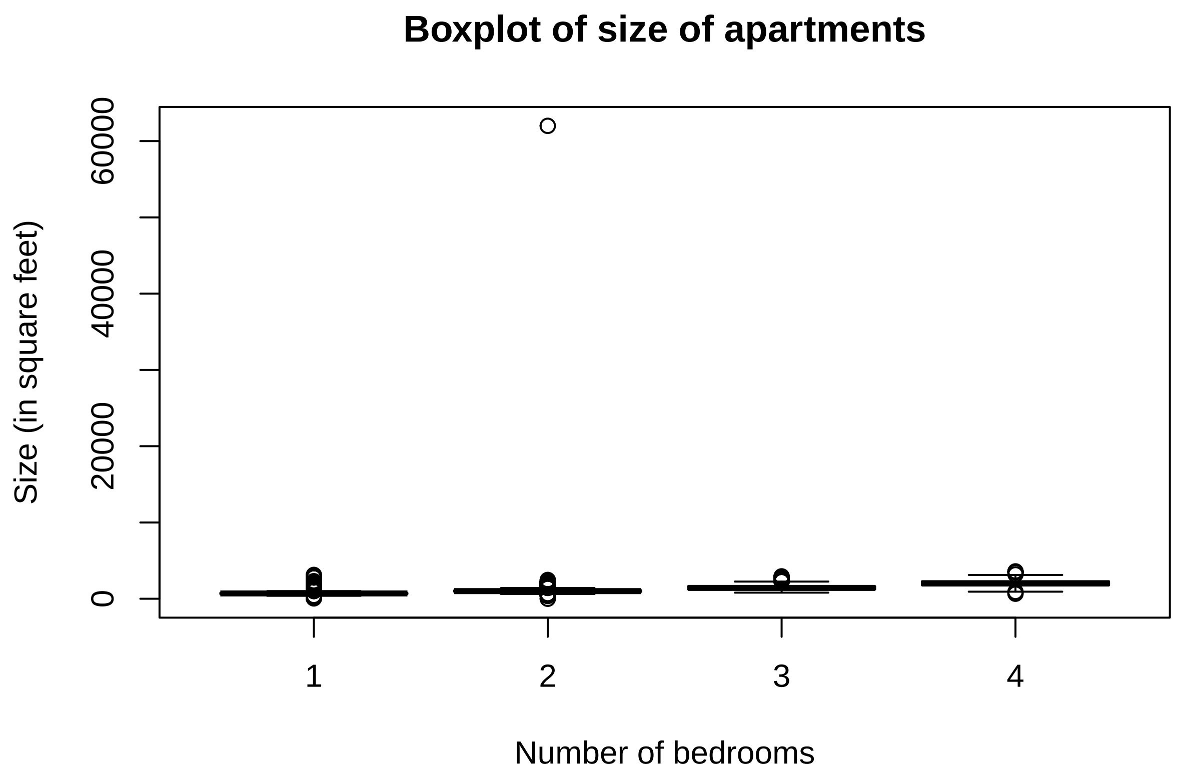
If we have a factor vector that separates the observations into different categories, we can create side-by-side boxplots. You do this by the formulation `y ~ fac`.

```
boxplot( craigslist$size ~ craigslist$brs ,
     main = "Boxplot of size of apartments",
     ylab="Size (in square feet)",
     xlab="Number of bedrooms"
)
```



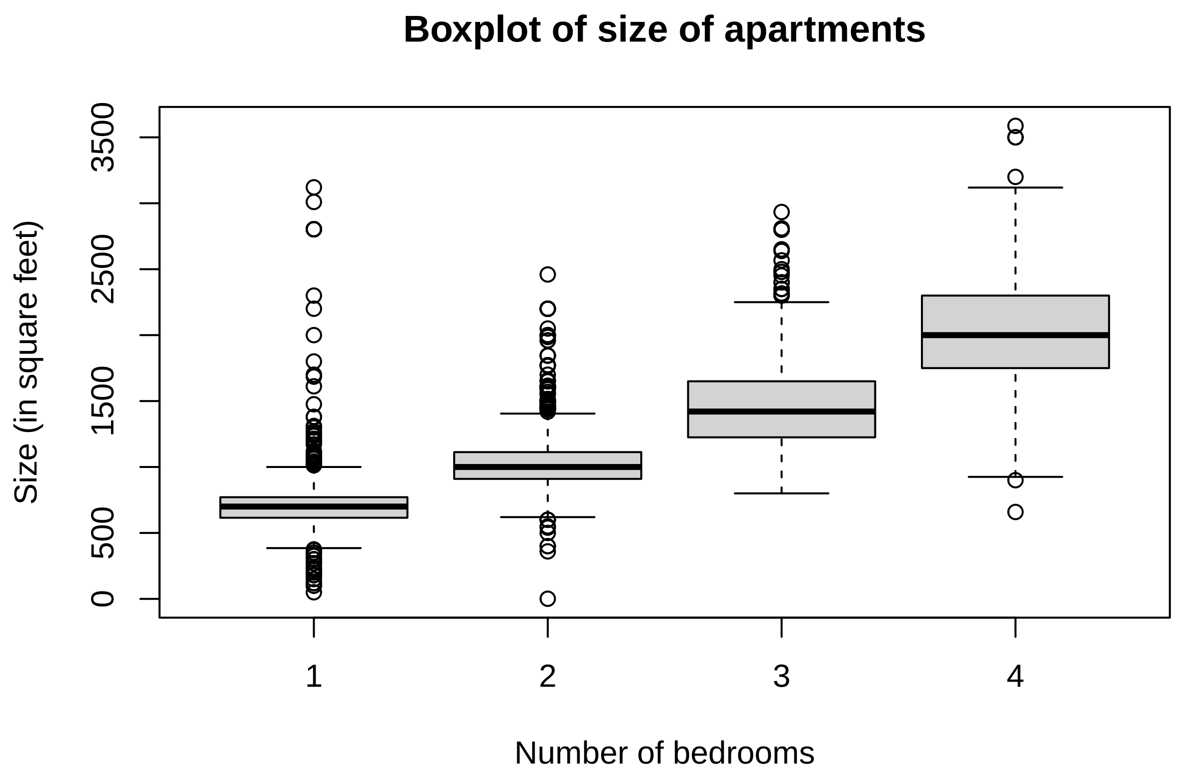
When both variables are stored in the same data.frame, we can also do the following (more elegant) method:

```
boxplot(size ~ brs, data=craigslist,
     main = "Boxplot of size of apartments",
     ylab="Size (in square feet)",
     xlab="Number of bedrooms"
)
```



Notice how this makes it easy to repeat the plot without that outlier by just subsetting the dataset given as input:

```
boxplot(size ~ brs, data=craigslist[craigslist$size<60000,],
     main = "Boxplot of size of apartments",
     ylab="Size (in square feet)",
     xlab="Number of bedrooms"
)
```


Boxplot of size of apartments
Size (in square feet)
0
500
1500
2500
3500
1
2
3
4
Number of bedrooms

Exercise 4.

(a) Subset to only the one bedroom postings.

```
# insert code here
```

(b) Draw the boxplot of one bedroom rent price for each cities side by side.

```
#Insert code here
```

(c) Can you find the link of the outlier with monthly rent larger than 5000 in the boxplot?

```
# insert code here
```

ggplot approach to plotting

The plotting commands we have used above are the built-in basic R graphic commands. They are fundamental building blocks that are useful to know. But another universe of plotting commands that have been created to create elegant plots easily. They are found in the library `ggplot2`, so let's load it (if you don't have `ggplot2`, you will need to install the package, see Lab 00)

```
library(ggplot2)
```

A 'ggplot2' plot is comprised of three fundamental building blocks:

1. Data: typically entered as a dataframe.
2. aesthetics (aes): to define which columns from the data will be used along which axes, and map the values to colors, symbol size, etc.
3. geom(s): to define the type of plot - point (geom_point), bars (geom_bar), lines (geom_line), etc.

'ggplot2' works in layers. We can create a base layer, and then add additional layers to it. New layers can be added using "+" operator.

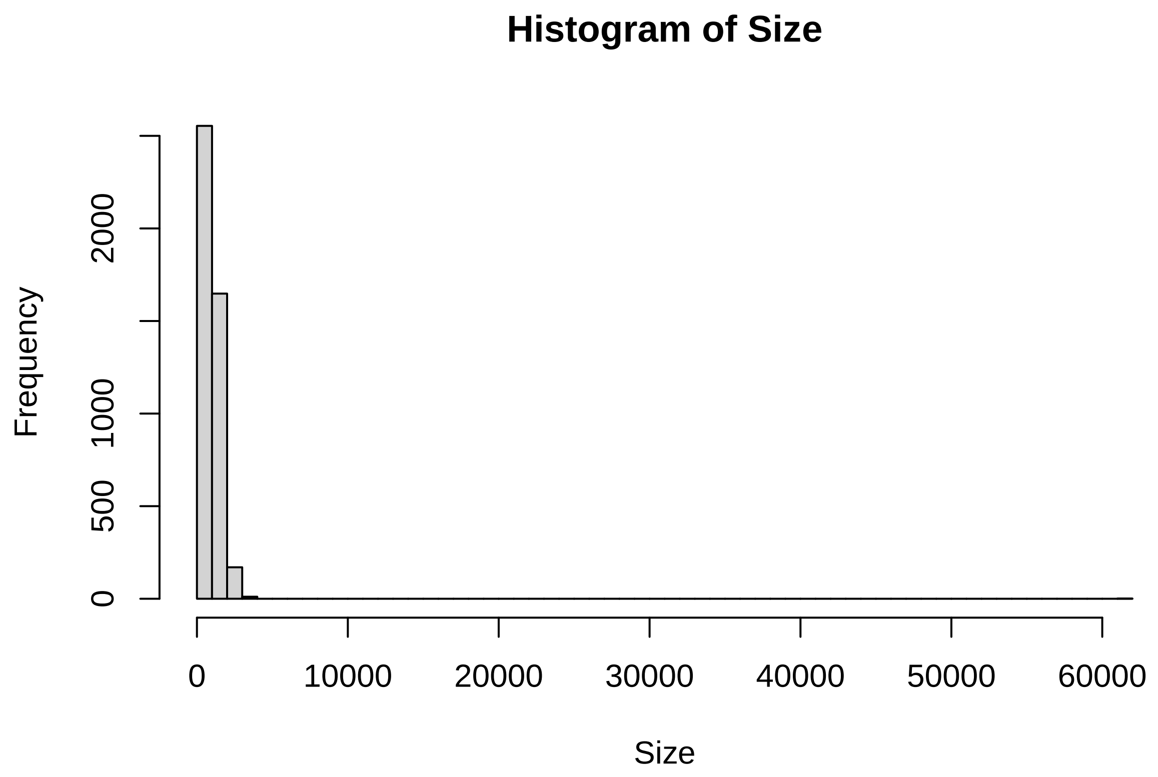
Every `ggplot2` plot has the following steps:

- Create a plot object with `ggplot()`
- Within `ggplot()` you will define the `data.frame` you want work with in the plot. Note that unlike basic R commands, `ggplot2` works solely on `data.frames`
- Map the aesthetics/features used to represent the data, e.g., the x, y locations, color, symbol size.
- Specify what graphics shapes, aka, geoms, to view the data, e.g., point, bars, lines. Add these to the `ggplot` with `+`. Examples are `geom_point`, `geom_bar` or `geom_line`. At least one `geom_XXX` has to be included.

Histograms

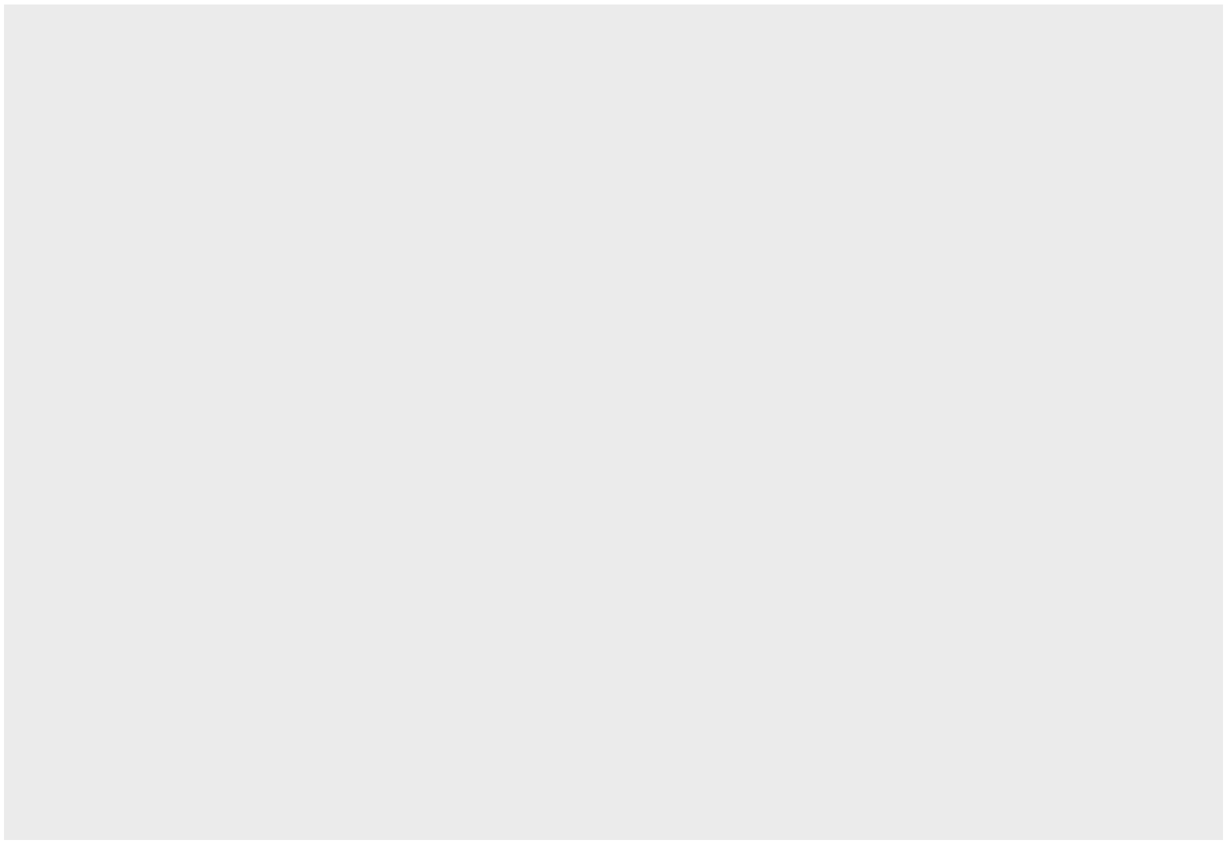
For example, we previously plotted with basic R:

```
hist(craigslist$size,
     main = "Histogram of Size", # plot title
     xlab = "Size", breaks=50)
```



With `ggplot2` we would have the following command to create the base layer

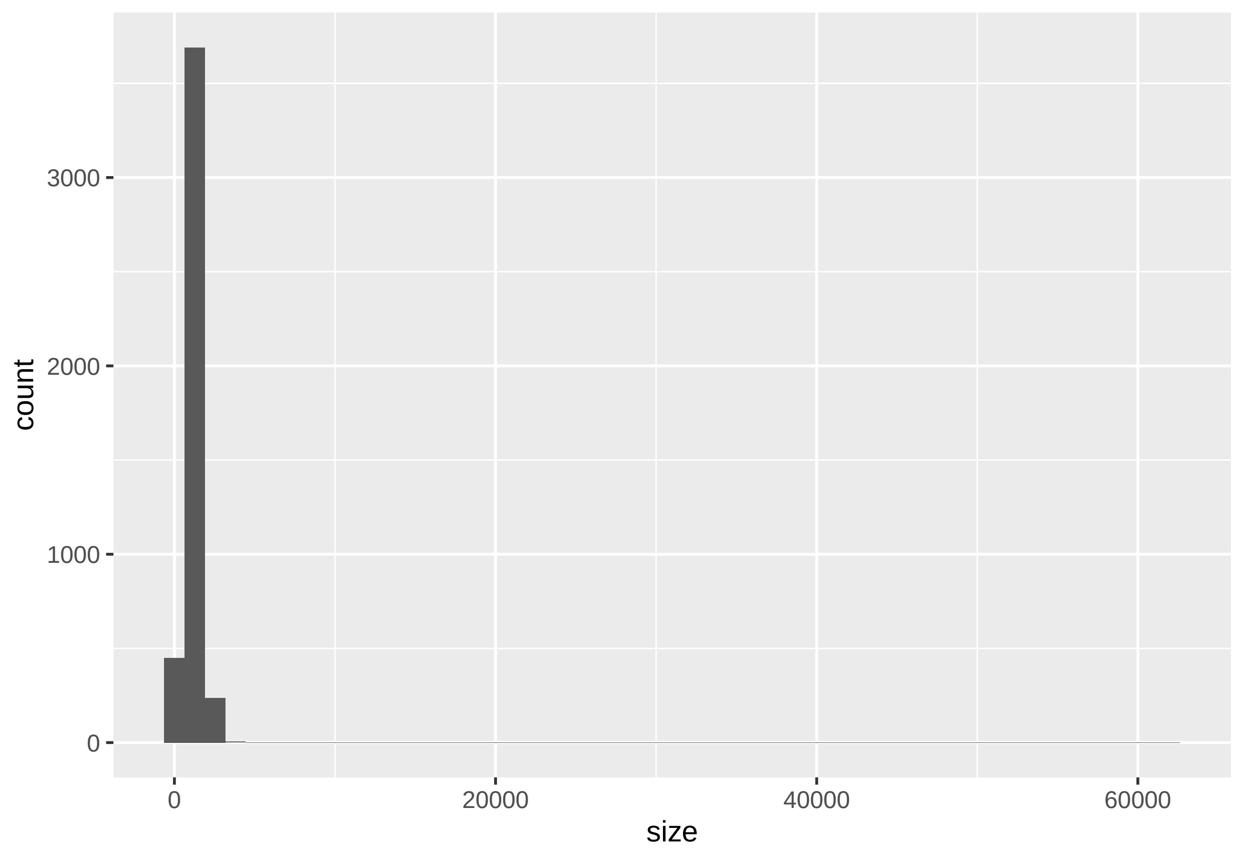
```
# ggplot version
ggplot(data = craigslist)
```



We can add the plot by specifying the kind of plot (histogram) using "+" operator to the base layer,

```
# ggplot version
ggplot(data = craigslist) + geom_histogram(aes(x = size),bins = 50)
```

```
## Warning: Removed 1492 rows containing non-finite values (`stat_bin()`).
```



`ggplot` provided the information about the data we are using. Then `geom_histogram` defines the plotting we want to do with the observations in our dataset. `aes(x = size)` are the aesthetics, providing the information about the column of the `data.frame` that forms the `x` axis data.

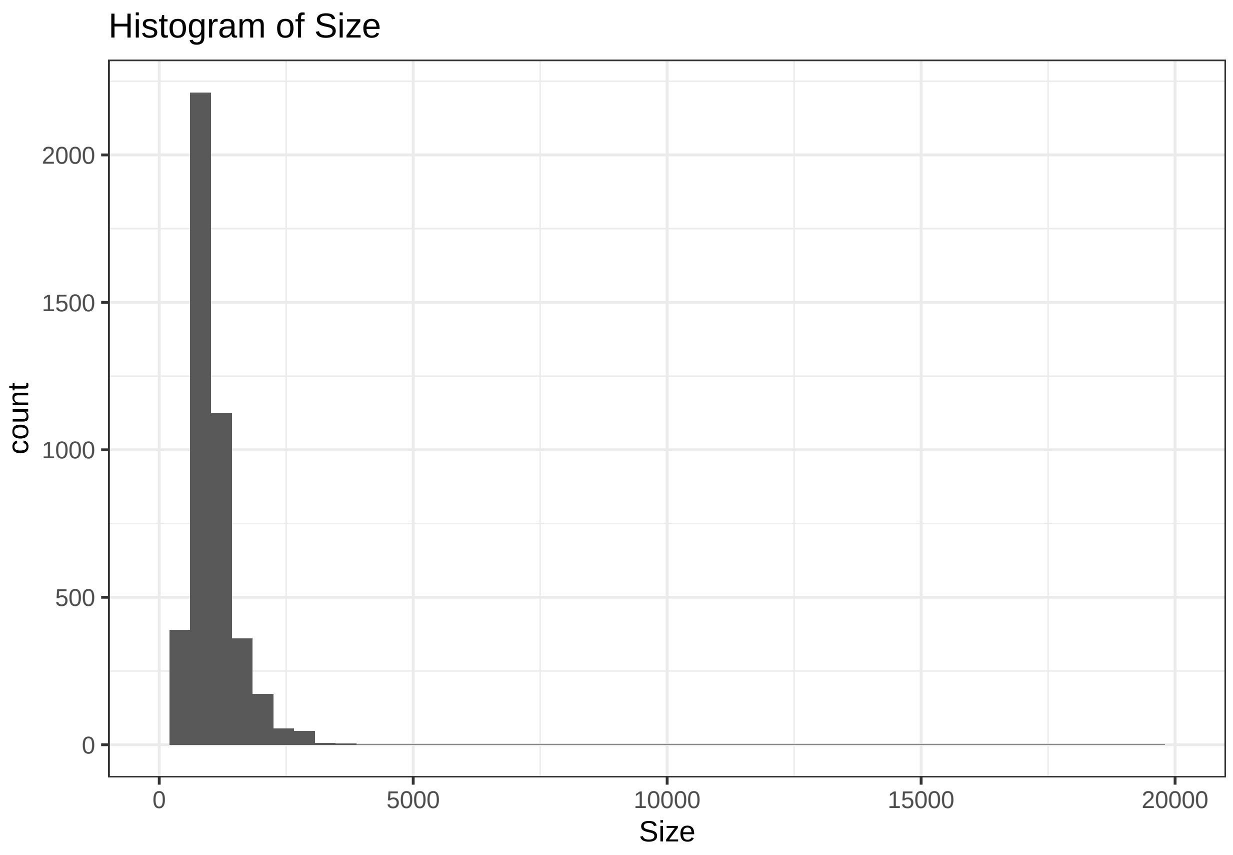
Notice that `ggplot` gives a warning about missing variables (`hist` just silently removed them without telling you).

We can add more features to our `ggplot`, just like with basic R plotting commands.

```
# ggplot version
ggplot(data = craigslist) +
  geom_histogram(bins = 50, aes(x = size)) +
  labs(x = 'Size', title = 'Histogram of Size') + #provide axis, plot labels
  xlim(0,20000) + # zoom in
  theme_bw() #change to black and white, instead of grey background
```

```
## Warning: Removed 1493 rows containing non-finite values (`stat_bin()`).
```

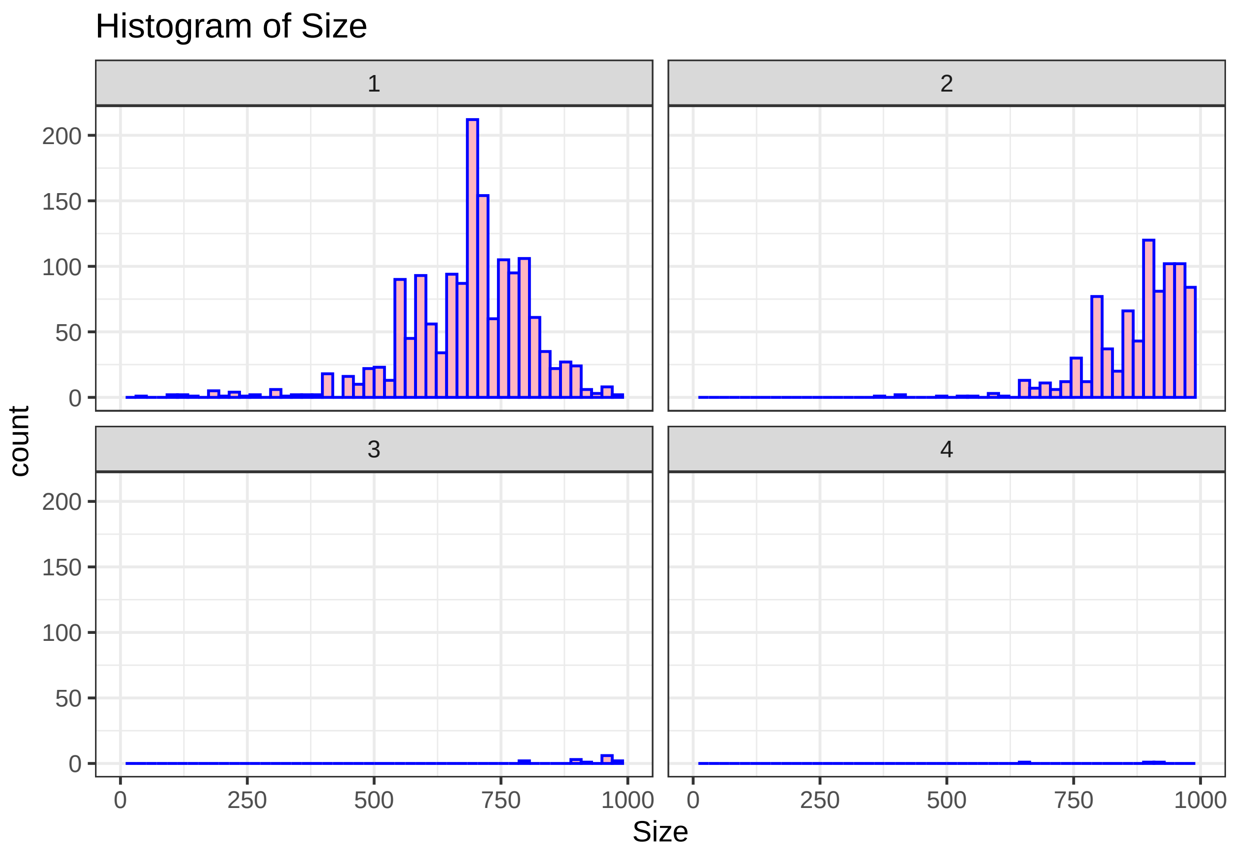
```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```



ggplot2 also makes it easy to make separate plots for subsets of the data (when those subsets are defined by a variable in your data.frame). These are called “facets” of the data

```
# ggplot version
ggplot(data = craigslist) +
  geom_histogram(bins = 50, aes(x = size), fill="light pink", color = "blue") +
  labs(x = 'Size', title = 'Histogram of Size') + #provide axis, plot labels
  xlim(0,1000) + # zoom in
facet_wrap(~factor(brs))+
  theme_bw() #change to black and white, instead of grey background
```

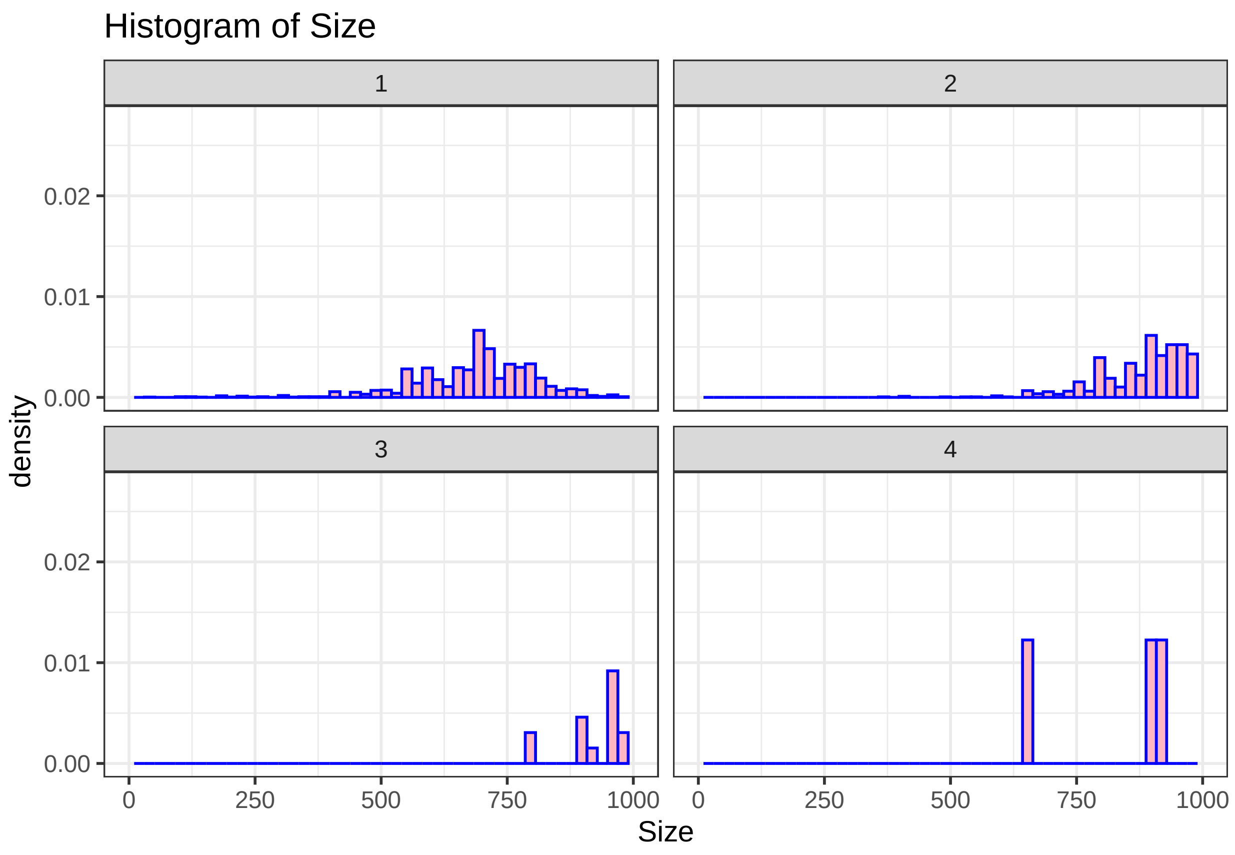
```
## Warning: Removed 3322 rows containing non-finite values (`stat_bin()`).

## Warning: Removed 8 rows containing missing values (`geom_bar()`).
```



We can also make density histograms of each group

```
# ggplot version
ggplot(data = craigslist) +
  geom_histogram(bins = 50, aes(x = size,..density..), fill="light pink", color = "blue") +
  labs(x = 'Size', title = 'Histogram of Size') + #provide axis, plot labels
  xlim(0,1000) + # zoom in
facet_wrap(~factor(brs))+
  theme_bw() #change to black and white, instead of grey background
```

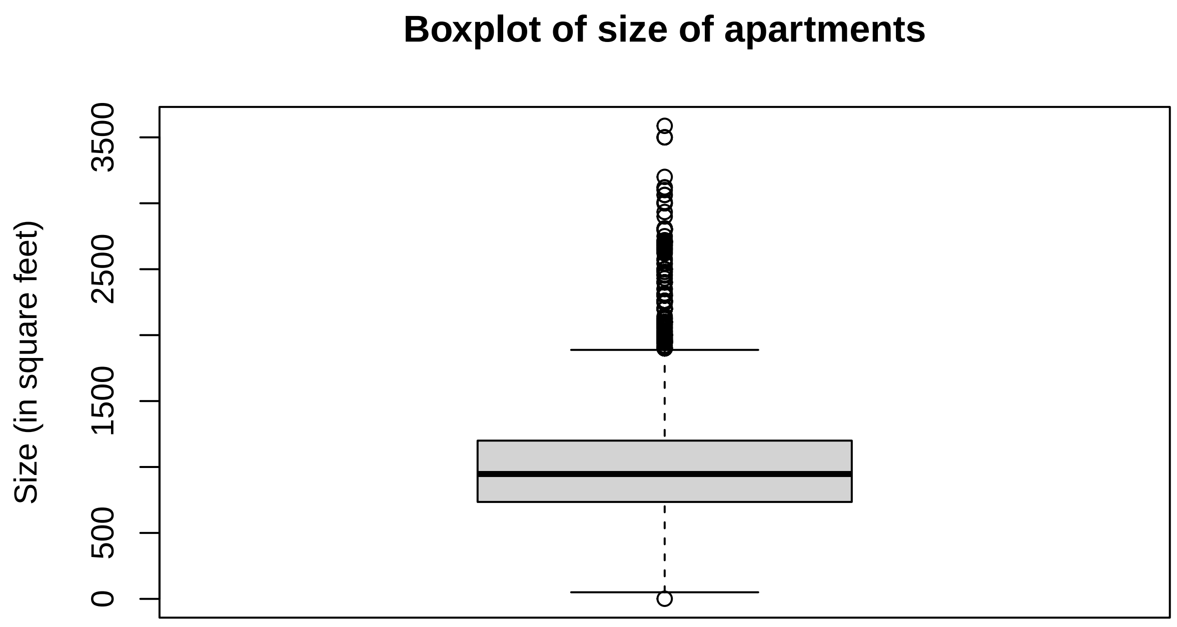
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.

## Warning: Removed 3322 rows containing non-finite values (`stat_bin()`).

## Warning: Removed 8 rows containing missing values (`geom_bar()`).
```



Boxplots

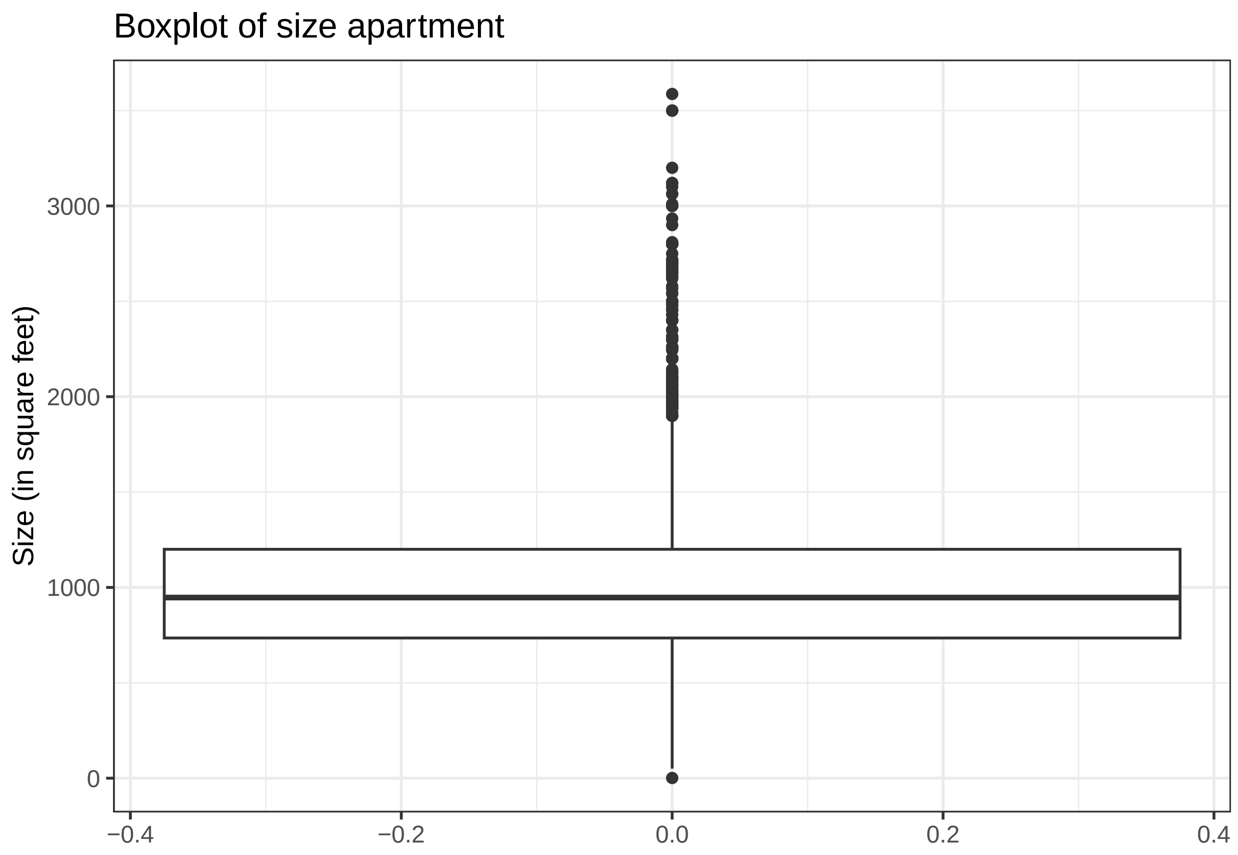
Here is our previous boxplot command,

```
boxplot(craigslist$size[craigslist$size<60000],
     main = "Boxplot of size of apartments",
     ylab="Size (in square feet)"
)
```



```
ggplot(craigslist[craigslist$size<60000,]) +
  geom_boxplot(aes(y = size)) +
  labs(y = 'Size (in square feet)', title = 'Boxplot of size apartment') +
  theme_bw()
```

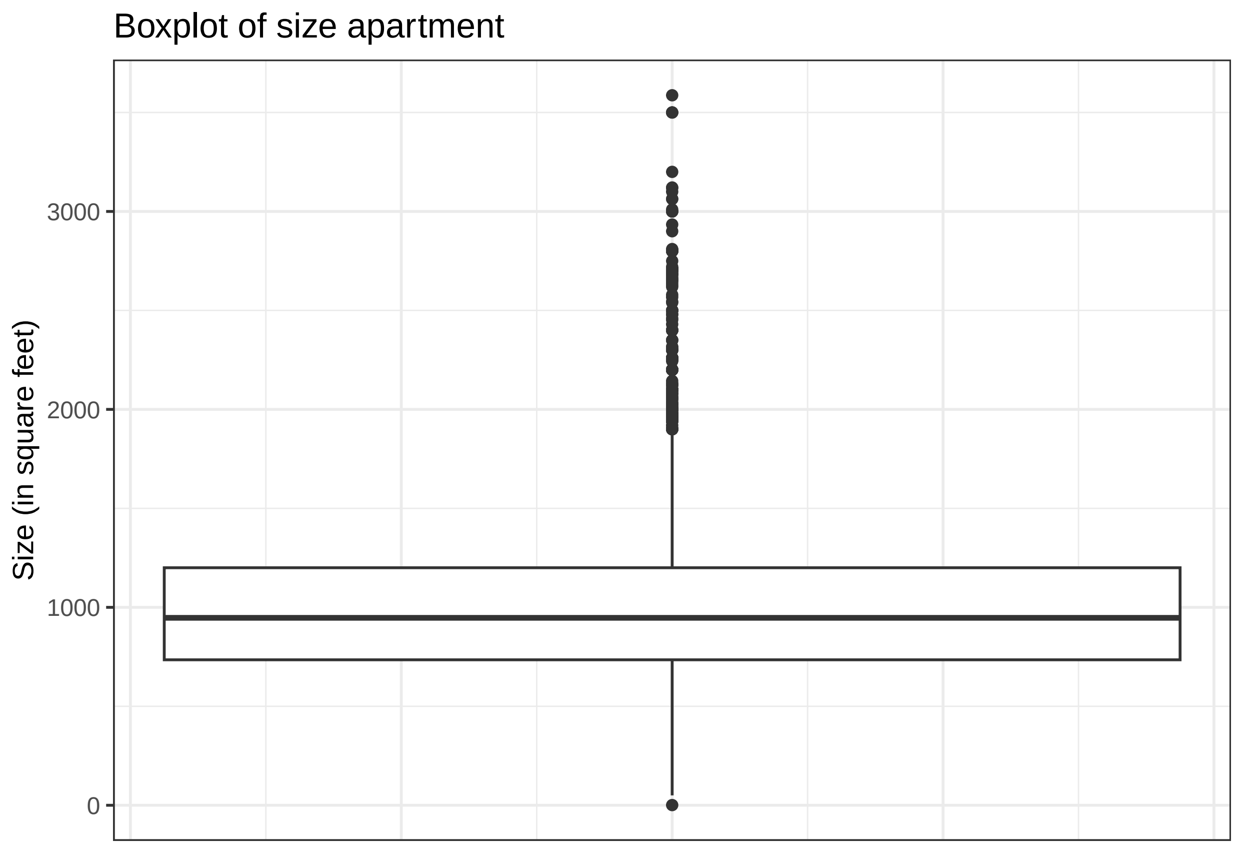
```
## Warning: Removed 1492 rows containing non-finite values (`stat_boxplot()`).
```



We don't really want the x-axis markings, so we can do this:

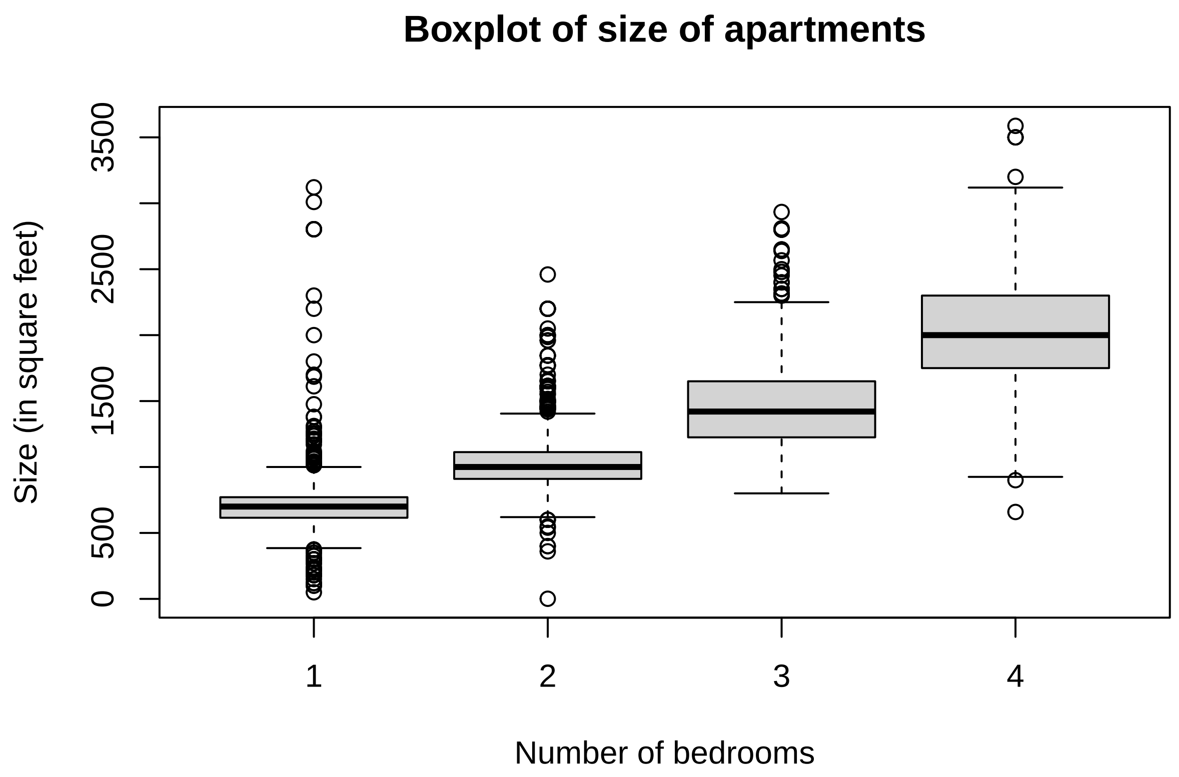
```
ggplot(craigslist[craigslist$size<60000,]) +
  geom_boxplot(aes(y = size)) +
  labs(y = 'Size (in square feet)', title = 'Boxplot of size apartment') +
  theme_bw() + # theme_bw needs to be before theme
  theme(axis.ticks.x = element_blank(), axis.text.x = element_blank()) # remove x axis ticks and labels
```

```
## Warning: Removed 1492 rows containing non-finite values (`stat_boxplot()`).
```



If we want to compare different groups we previously had

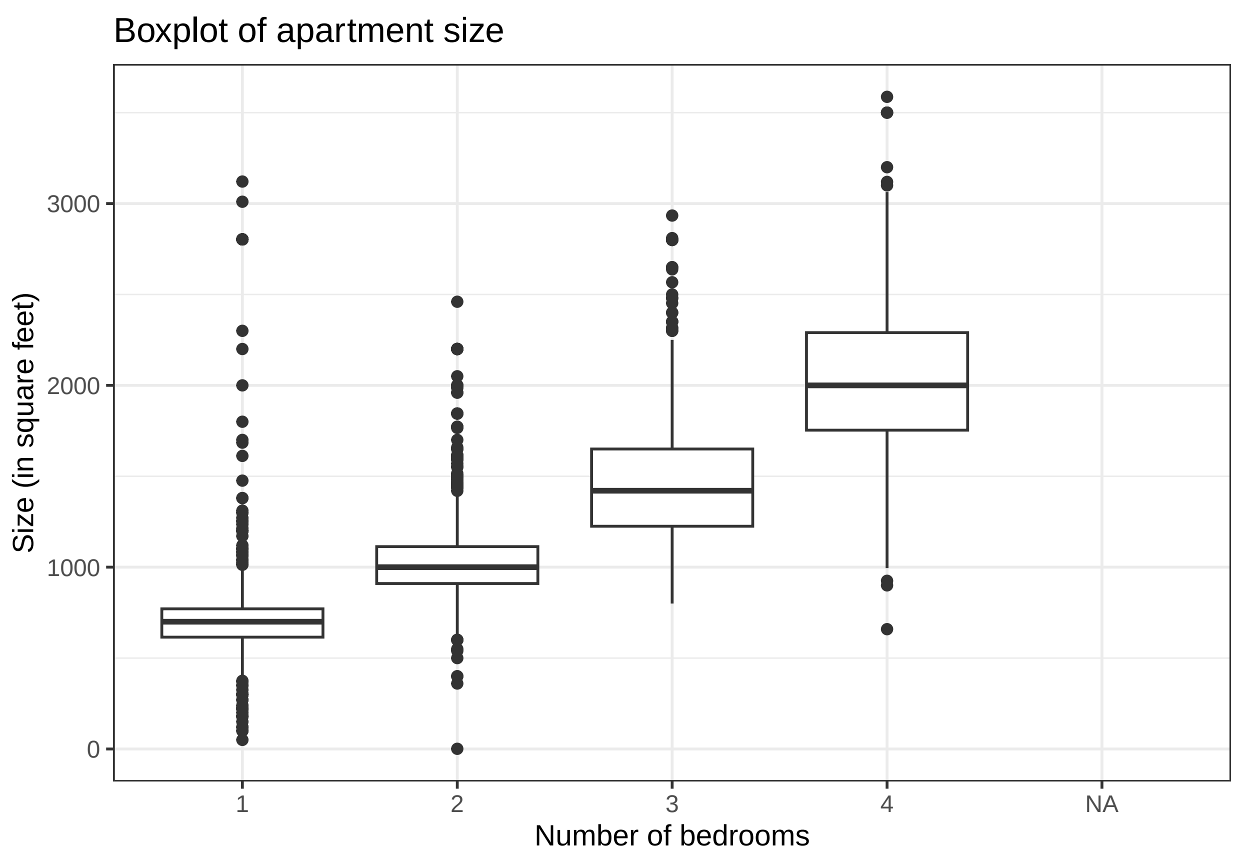
```
boxplot(size ~brs , data=craigslist[craigslist$size<60000,],
     main = "Boxplot of size of apartments",
     ylab="Size (in square feet)",
     xlab="Number of bedrooms"
)
```



To do the same thing in `ggplot` we would define our `x` variable to be the variable that we want to separate them by. The variable we need to split them by needs to be a `factor`, and number of bedrooms is numeric, so we use the `factor` function to make it treat it as a factor.

```
ggplot(craigslist[craigslist$size<60000,]) +
  geom_boxplot(aes(x = factor(brs), y = size)) +
  labs(x = 'Number of bedrooms',
       y = 'Size (in square feet)',
       title = 'Boxplot of apartment size') +
  theme_bw()
```

```
## Warning: Removed 1492 rows containing non-finite values (`stat_boxplot()`).
```

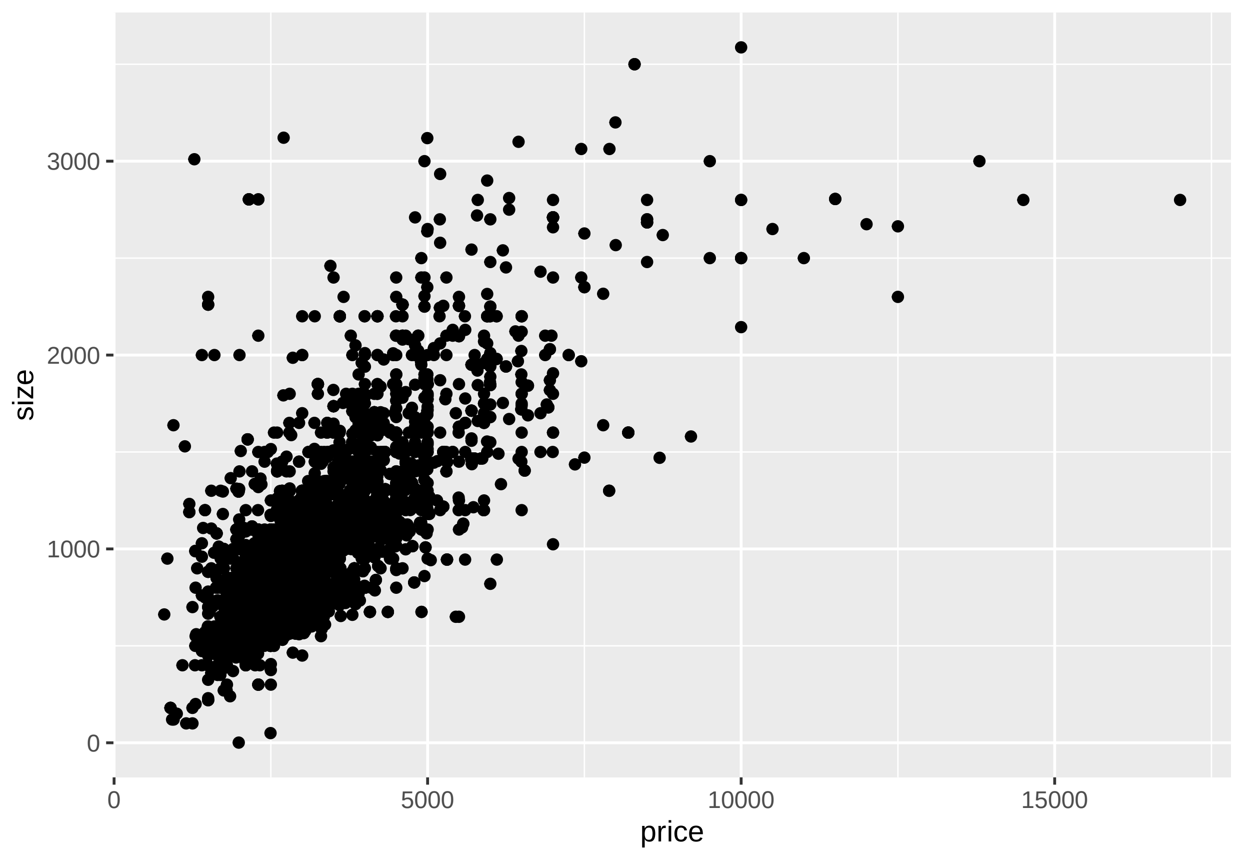


Scatterplots

`ggplot2` also makes scatter plots using `geom_point`.

```
ggplot(craigslist[craigslist$size<60000,]) + geom_point(aes(x=price,y=size))
```

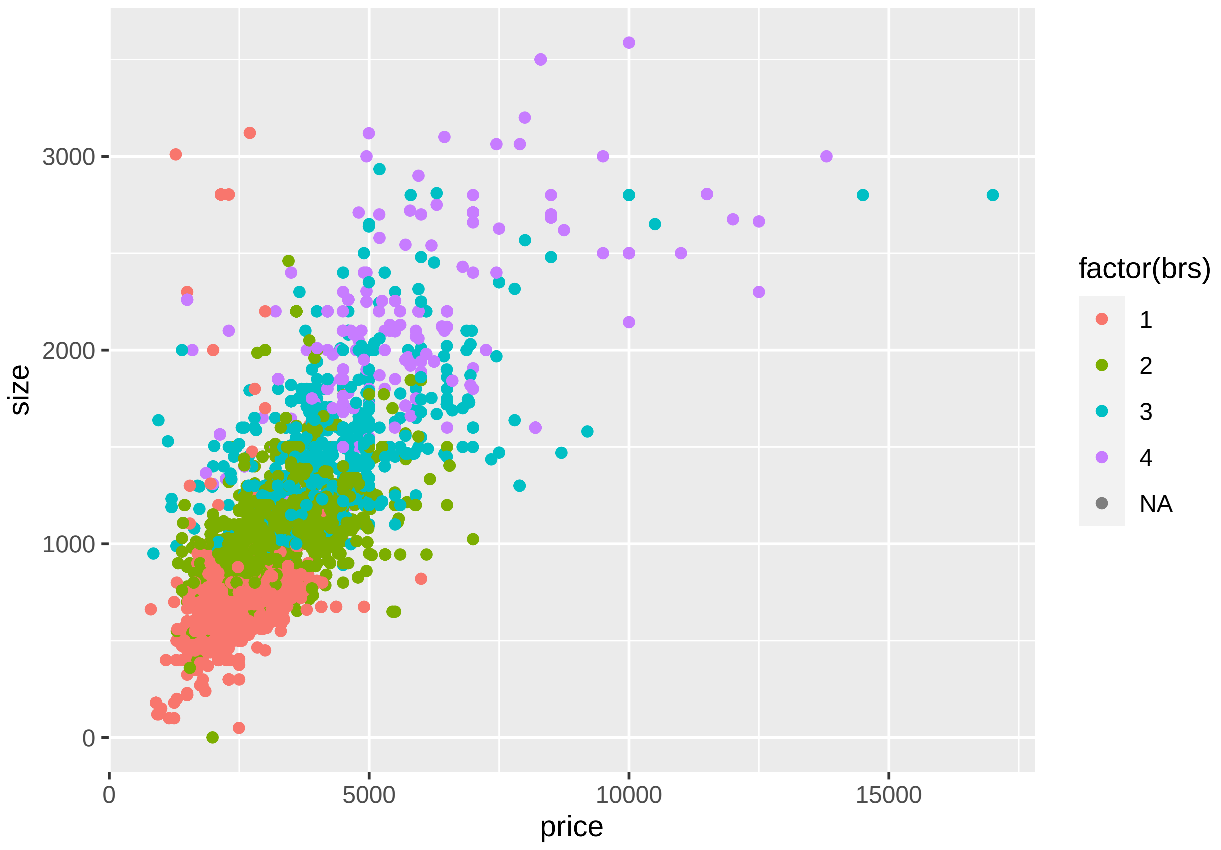
```
## Warning: Removed 1492 rows containing missing values (`geom_point()`).
```



We can easily color the points by number of bedrooms

```
ggplot(craigslist[craigslist$size<60000,]) + geom_point(aes(x=price,y=size, color=factor(brs)))
```

```
## Warning: Removed 1492 rows containing missing values (`geom_point()`).
```



Exercise 5.

Make separate scatterplots of size vs price for each location; on each scatterplot color the points by the number of bedrooms, as above. Give a title to your plot and informative (non-default) x and y axis labels.

```
# insert code here
```