Exercise 9.8 - Walkthrough

1 Data Import

We begin as always, by including the relevant libraries. In our case, that's none other than the 'GG plot 2' library. In addition we incorporate the 'wesanderson' library from which we'll choose a color palette later on.

```
1  library("ggplot2")
2  library("wesanderson")
3
```

The next step is to load the data for the task. It is located in the 'real esate CSV' and you can download it with the resources for this lecture.

As it contains information about 'California real estate properties', let's load it onto a variable with the name 'real estate'. Don't forget to specify the path to your file using forward slashes. Also, to include the column names from our 'CSV', let's set header to true and add a comma as a separator.

```
4  df_real_estate <- read.csv("real_estate.csv",
5                             header = TRUE,
6                             sep = ",")
```

Before we proceed with plotting, we might want to take a quick look at our new data frame.

We have several columns, such as building id and building type. However, for our scatter plot, we'll rely on the 'Area' and 'Price'.

	ID	Building.Type	Year.of.sale	Month.of.sale	Type.of.property	Property..	Area..ft..	Price	Status
1	1030	1	2005	11	Apartment	30	743.09	246.1727	Sold
2	1029	1	2005	10	Apartment	29	756.21	246.3319	Sold
3	2002	2	2007	7	Apartment	2	587.28	209.2809	Sold
4	2031	2	2007	12	Apartment	31	1604.75	452.6670	Sold
5	1049	1	2004	11	Apartment	49	1375.45	467.0833	Sold
6	3011	3	2007	9	Apartment	11	675.19	203.4918	Sold
7	3026	3	2007	9	Apartment	26	670.89	212.5208	Sold
8	3023	3	2008	1	Apartment	23	720.81	198.5918	Sold
9	1031	1	2006	6	Apartment	31	782.25	265.4677	Sold

2 Creating the scatter plot

We begin by creating a 'scatter' 'GG plot'.

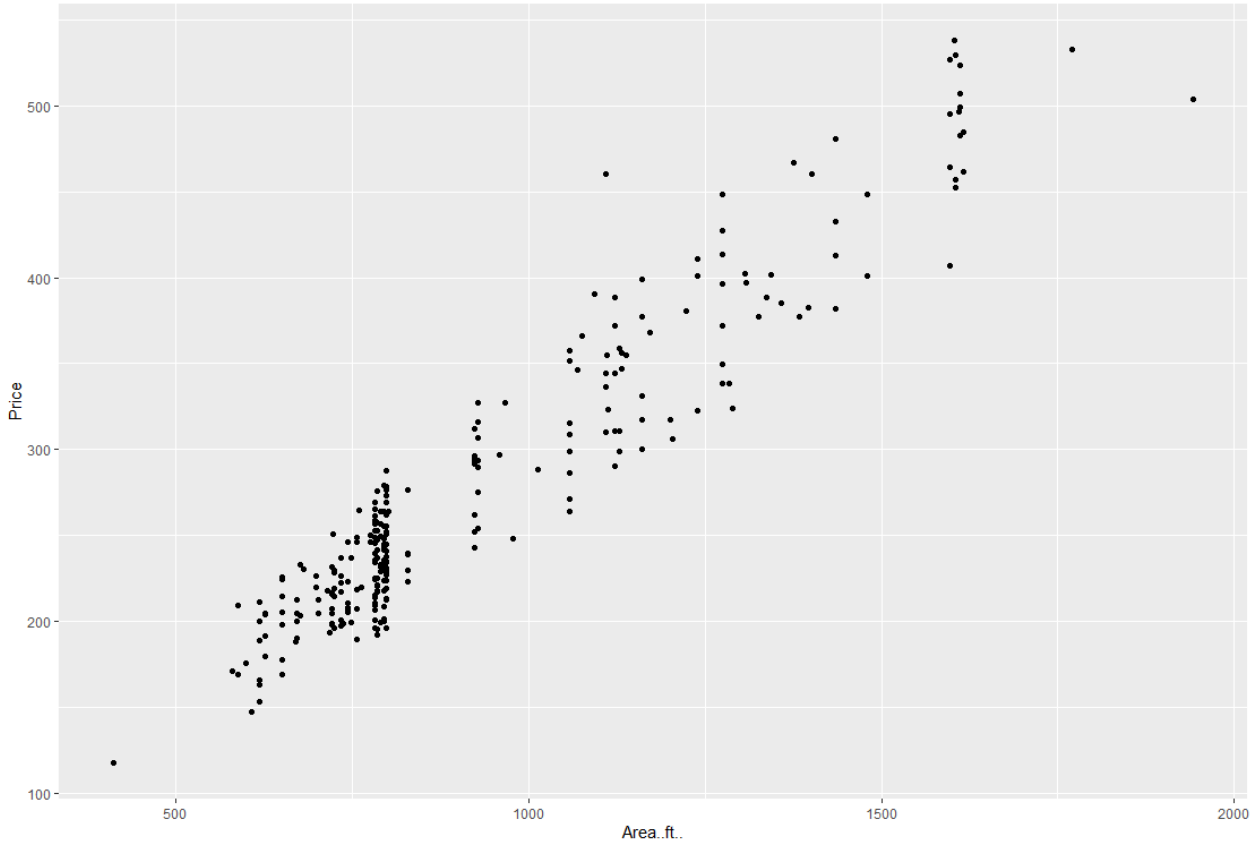
```
7  scatter <- ggplot(df_real_estate,
8                    aes(x = Area..ft..,
9                        y = Price)) +
```

The first argument is our 'Real Estate' data set. What follows are the aesthetics. We'll be plotting 'Area' versus 'Price', so x will be 'Area' and the y-axis is going to display the 'Price' in Feet.

Cool.

Onto the 'geometry'. To create a scatter plot using 'GG plot' we should specify the geometry as a 'geom point'.

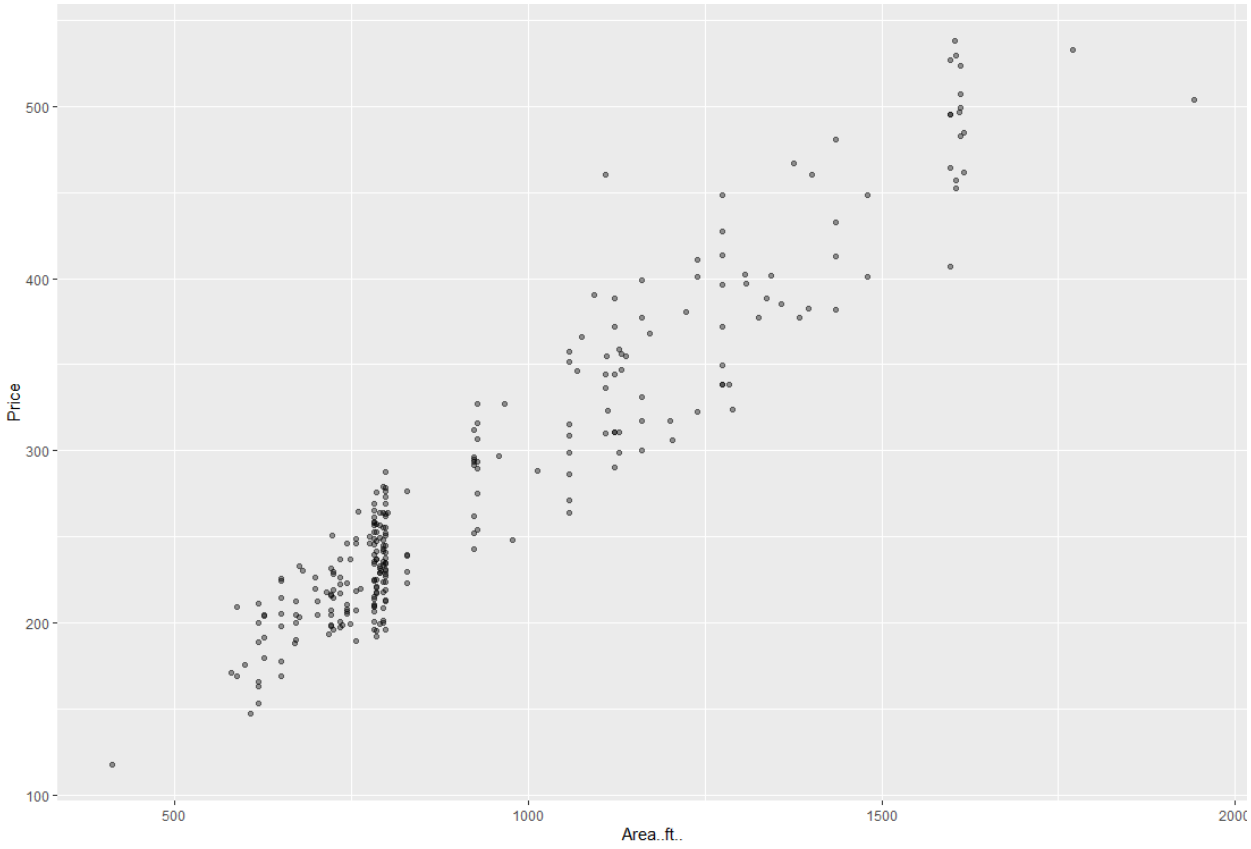
And that's all three required layers dealt with.



A scatter plot shows each point from the data, so, if we have a large data set, this might result in overplotting. In other words, we could have too many points jumbled together or even on top of each other. To avoid this issue, we can add a transparency level.

In R, transparency is controlled with the 'alpha' parameter of our 'geom point'. It takes values between 0 and 1. For our chart, an alpha of 0.4 should suffice.

This is the result:



3 Including an additional color feature to the scatter plot

Now, one of the advantages of a scatter plot, is that we can color code according to a third variable. In this way, we're able to incorporate much more information in a scatter plot. Something you may find crucial when conducting your own analysis.

So, let's see how to include color coding with 'GG plot'. We're going to need an additional color argument, as well as a variable on which to color. In our case, it's the 'Building Type'.

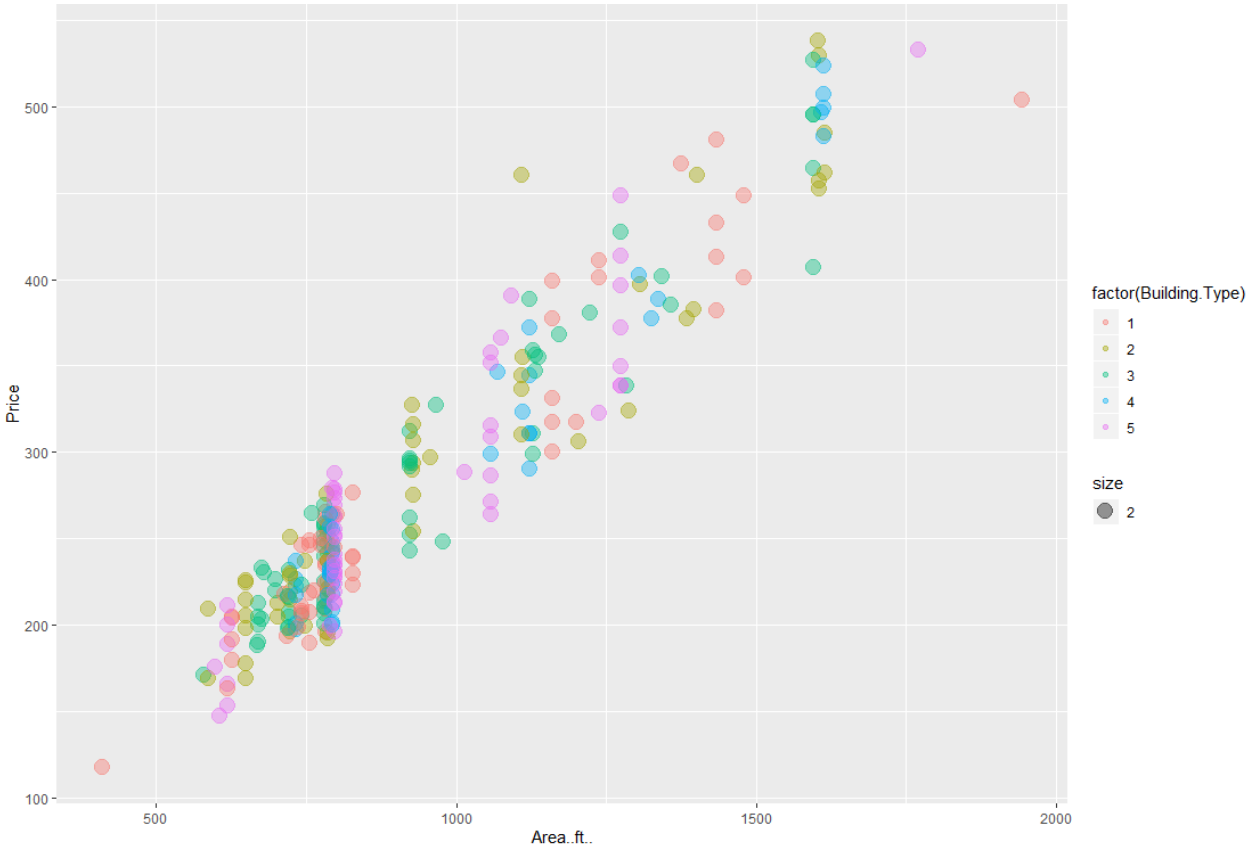
In 'GG plot' lingo, this means first adding a 'color' 'aesthetic' for the 'geometric point'. We're coloring based on the 'Building' type, so that's what our color has to be.

Next, to have the 'building type' values as strings, we need to use the factor function.

As a last detail, let's increase the size of each point to a '2'.

```
10                geom_point(aes(color = factor(Building.Type),
11                               size = 2), #size of the points on the scatter
12                               alpha = 0.4)
```

And let's see the result!



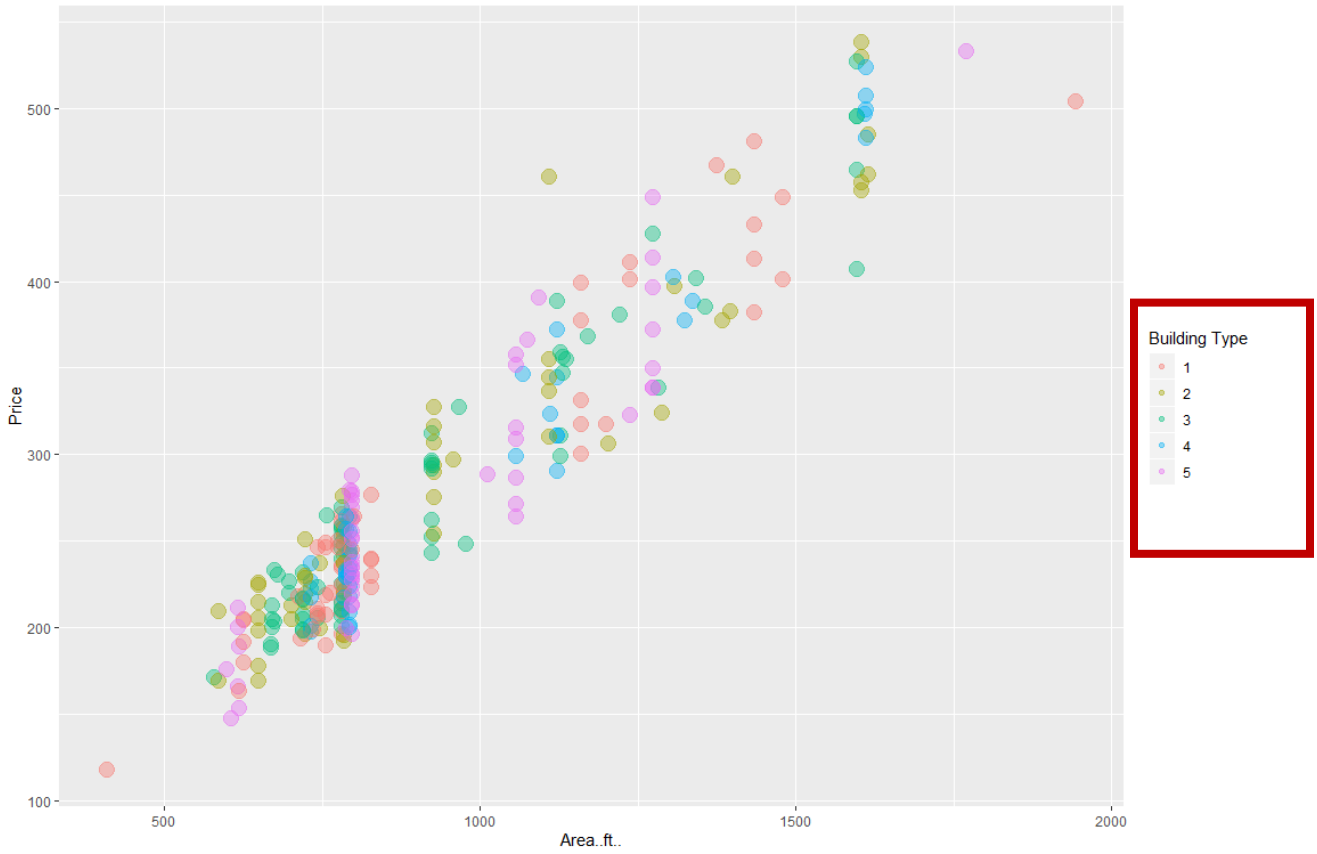
4 Chart styling and formatting

Before we move on, though, let's format the legend a little.

Beginning with removing the legend for size. We can do this with guides and by setting the size to false.

Then, we need to change the name for the top legend. Let's use 'labs' as short for labels. There we'd need to refer to the color argument and set its name to 'Building Type'.

```
13                  guides(size = FALSE) +
14                  labs(color = "Building Type")
```



Since our chart has become more colorful, we could discuss picking an alternative color palette.

In this lecture, we'll use a movie inspired color palette.

A favourite of mine is the 'R' 'Wes Anderson' library, which contains color palettes inspired by his films.

```
25  names(wes_palettes)
```

If you'd like you can take a glimpse at the palettes this package offers. We simply type 'Names' and in brackets 'Wes underscore Palettes'.

```
> names(wes_palettes)
 [1] "BottleRocket1"  "BottleRocket2"  "Rushmore1"      "Rushmore"       "Royal1"         "Royal2"         "Zissou1"        "Darjeeling1"
 [9] "Darjeeling2"    "Chevalier1"     "FantasticFox1"  "Moonrise1"      "Moonrise2"      "Moonrise3"      "Cavalcanti1"    "GrandBudapest1"
[17] "GrandBudapest2" "IsleofDogs1"    "IsleofDogs2"
>
```

We can choose from almost 20 Wes Andersen movie palettes, including palettes for the movies 'Rushmore', 'Moonrise' and 'Darjeeling'.

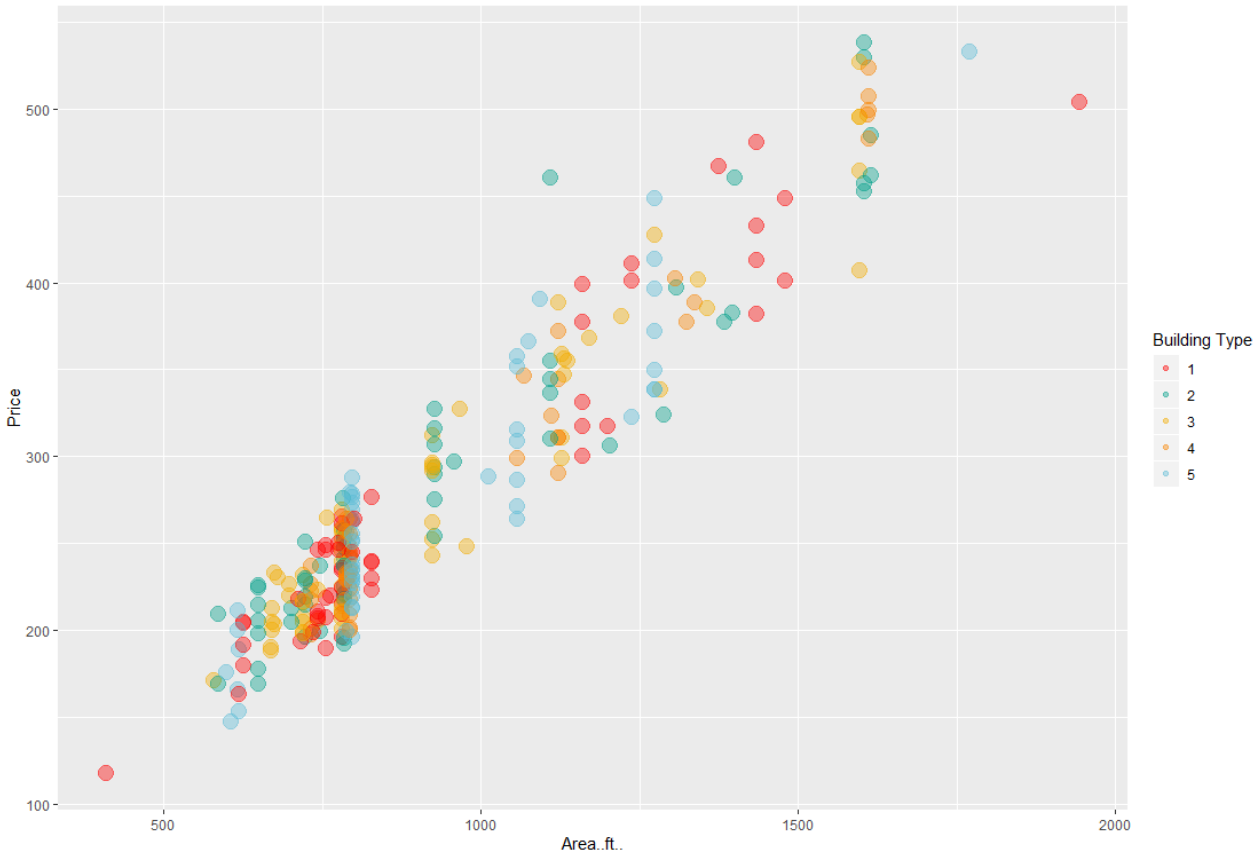
Keep in mind here, that most palettes have only 4 or 5 colors, which might not be enough, depending on your data.

Let's add a new color palette to our graph.

We can achieve this through the 'Scale Color Manual' function. In the brackets, as values, let's specify the name of the palette and the number of colors we require. Our palette is 'Darjeeling 1' and, as mentioned, we need five colors, one for each of the building types.

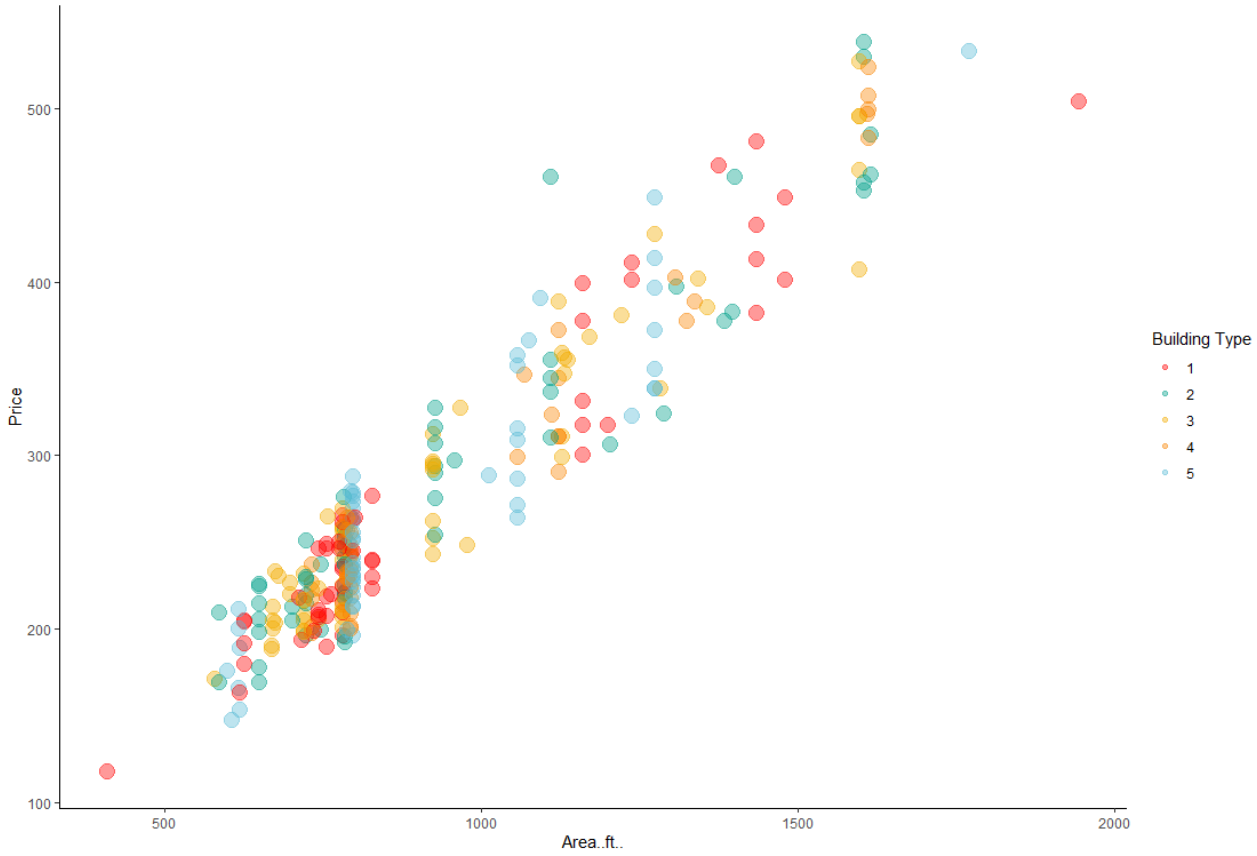
```
15          scale_color_manual(values = wes_palette(name = "Darjeeling1",
16                                                  n = 5))
```

Let's see the result!



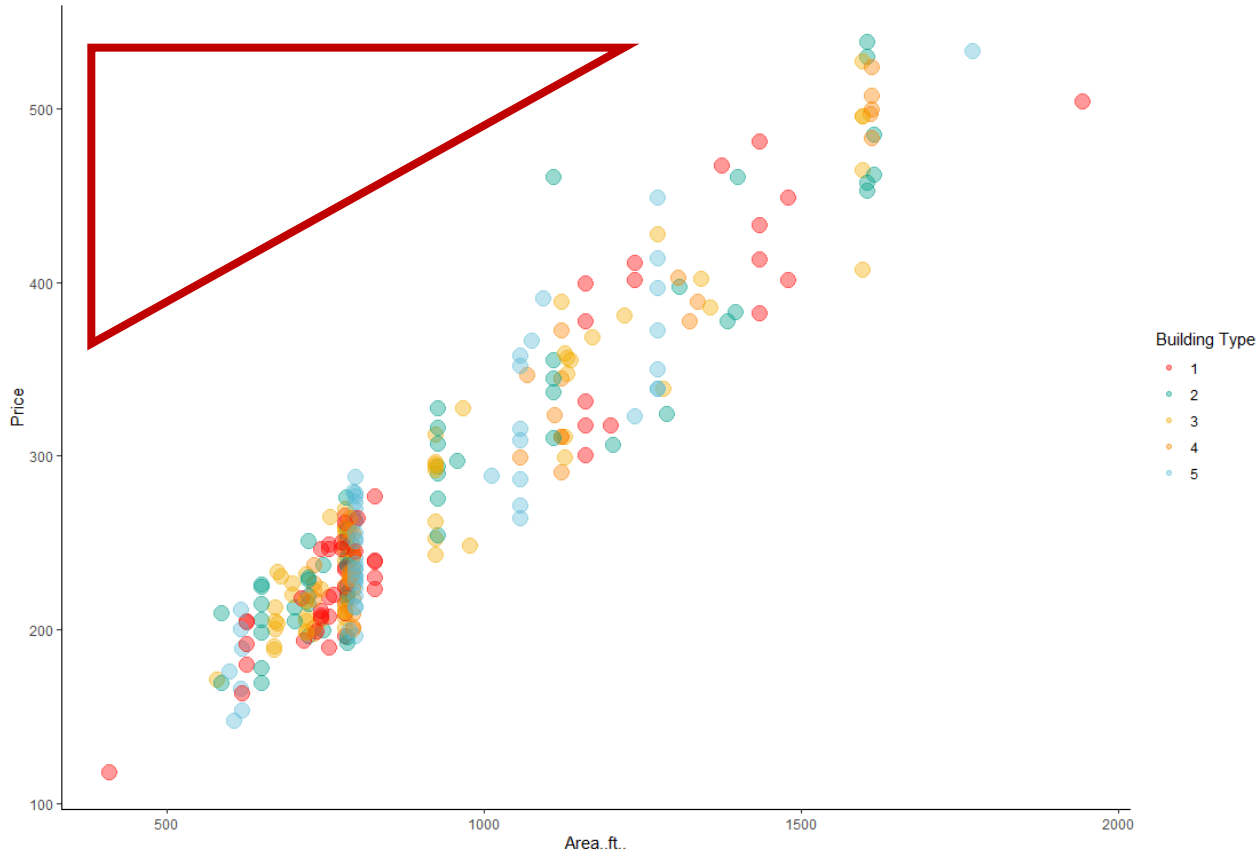
Let’s include a custom theme, which will be theme_classic.

```
17          theme_classic()
```



Next, we can change the position of the legend to increase space for our plot.

A better place for the legend will be the top left corner of the graph. This way, we're taking advantage of the free space in the area highlighted by the red triangle.



To move the legend we must include a theme. There we'll specify the justification and position of the legend. Both are 0.01 on the x-axis and 1 on the y-axis.

```
18          theme(legend.justification = c(0.01, 1),
19                legend.position = c(0.01,1))
```

Lastly, let's not forget adding a title or should I say, a 'GG title'?

In any case, a nice name would be 'Relationship between Area and Price of California Real Estate'.

While we're at it we might address the axis labels. With 'x lab' set to 'area in square feet' and 'y lab' as 'price in thousands of dollars', we should be all set.

```
20          ggtitle("Relationship between Area and Price of California Real Estate") +
21          ylab("Price (000's of $)") +
22          xlab("Area (sq. ft.)")
```

5 Interpretation

It's time for the interpretation lecture of the scatter plot.

We used information on 'California real estate' housing and created a scatter plot using two of the property's features. On the x-axis, we have the area in ft, while the y-axis displays the price.

Now, for each point the scatter we can observe the values for both features. We can check the value of a point, for instance, the one on the far right and top of the points from the table and see its values. It has an area of 1942.5 and a price of over 500,000 dollars. If we'd like to, we could check the precise values for multiple points on the chart.

This is by far the most expensive and also the largest property. Which is clear, as the scatter plot shows each point in relation to the remaining data.

So, through the scatter plot, we were capable of plotting every observation from our data set, based on its price and area.

But more importantly, we managed to show the relationship between our two numerical features. And what we observe is that there is positive collinearity between size and property. Meaning that the larger the size of the property, the higher its price tends to be. Which of course makes sense, as larger properties tend to cost more.

While we can make many presumptions, we should not forget that there are other factors at play. The location of a property is also very important. Therefore, a single scatter plot does not contain all the answers.

Finally, in a scatter plot, the entire data set could be seen at once, which gives us a great advantage. Apart from the interrelation of the two variables, we can also notice each observation or point on the chart. This means we can easily spot when points are further away from the main group as they don't share the same properties or don't follow the general trend. We call such points **outliers.**

Does our chart contain such points?

Well, it seems that this point over here doesn't reflect the pattern of the remaining points. It's so far away, that it begs us to question the integrity of the data point. It is possible that either the price or the area of this property was entered incorrectly. If we come back to the point on the right-hand side of the plot we observe, we'll see that even though it's further away from the main body of points, it still reflects the main relationship between variables.

These types of observations are easily gained with the help of a scatter plot. However, it would be next to impossible to make with other charts, or with the raw data.

We will revisit this chart when we reach the exploratory data analysis section of the course!