ggplot2 Series 1 - Scatterplots

By: Thalia

```
# Load libraries
library(ggplot2)
library(ggfortify)
library(gridExtra) # Use to arrange the ggplots
```

ggplot only wok with dataframes but not individual vectors. All the data needed to make the plot is contained within the dataframe. The dataset we will use for this tutorial is from Kaggle: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

```
# Load data for demonstration
house_data <- read.csv('house_price.csv', header = TRUE)
head(house_data[c('GrLivArea', 'OverallCond', 'SalePrice', 'GarageType')], 3)
```

```
##   GrLivArea OverallCond SalePrice GarageType
## 1      1710           5    208500     Attchd
## 2      1262           8    181500     Attchd
## 3      1786           5    223500     Attchd
```

Columns we will use in this tutorial:

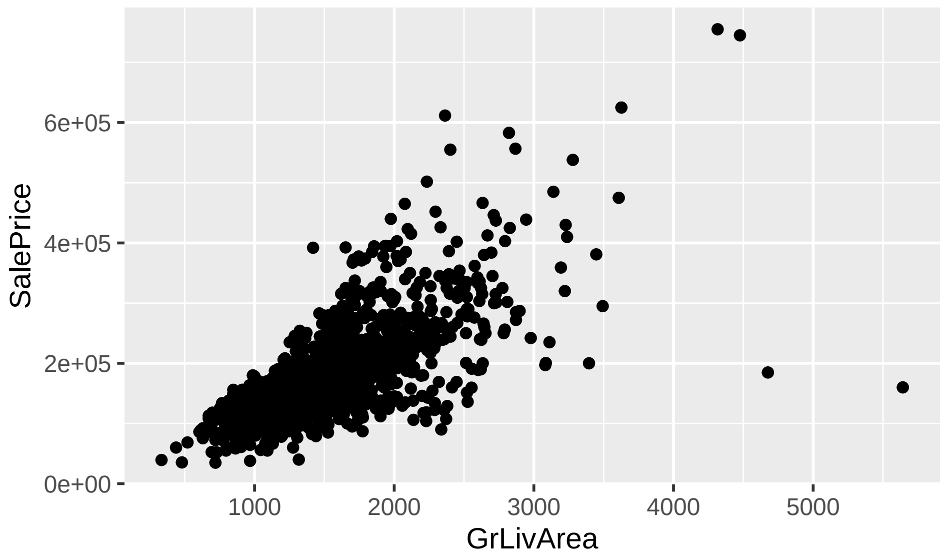
- GrLivArea: Above grade (ground) living area square feet
- OverallCond: Overall condition rating
- SalePrice: the property's sale price in dollars
- GarageType: Type of Garage location

Basic Syntax

```
# Initialize ggplot
gg_init <- ggplot(house_data, aes(x=GrLivArea, y=SalePrice))
```

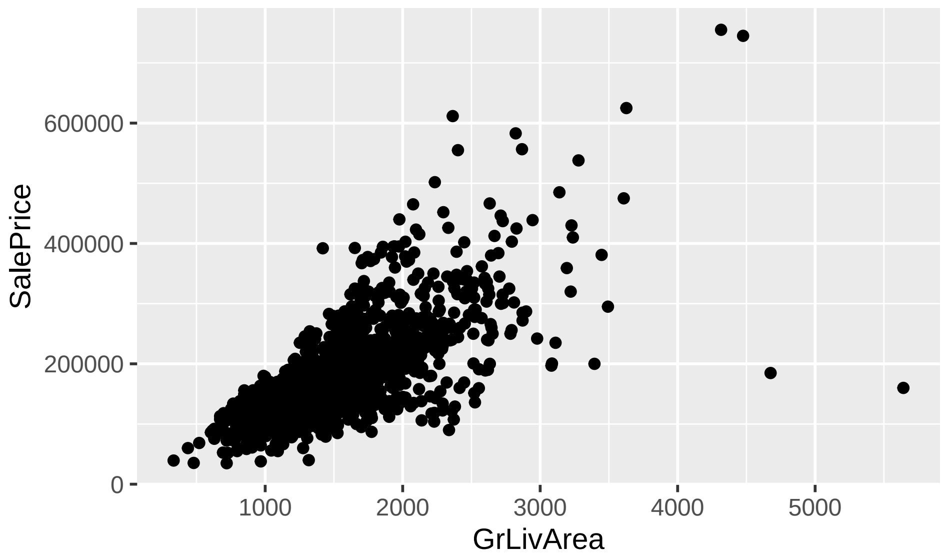
Note ggplot itself does not generate any plot. If you print out the ggplot object we just defined, it will draw an blank plot. with **aes** we told ggplot what is the x and y axis we want to use for our plot. To make a scatterplot, we need to add an geom layter on top of the initial blank plot.

```
# Basic syntax of scatterplots
gg_init + geom_point()
```



Let's turn off the 1e+06 notation

```r
options(scipen=999)  # turn off scientific notation like 1e+06
gg_init + geom_point()
```



Now we get a scatter plot. We can further customize it.

Change the shape of dots

```r
# Basic syntax of scatterplots
gg1 <- gg_init + geom_point(shape=1)
gg2 <- gg_init + geom_point(shape=5, size=2)
gg3 <- gg_init + geom_point(shape=17, size=1.5, colour='steelblue')
grid.arrange(gg1, gg2, gg3, ncol=3)
```

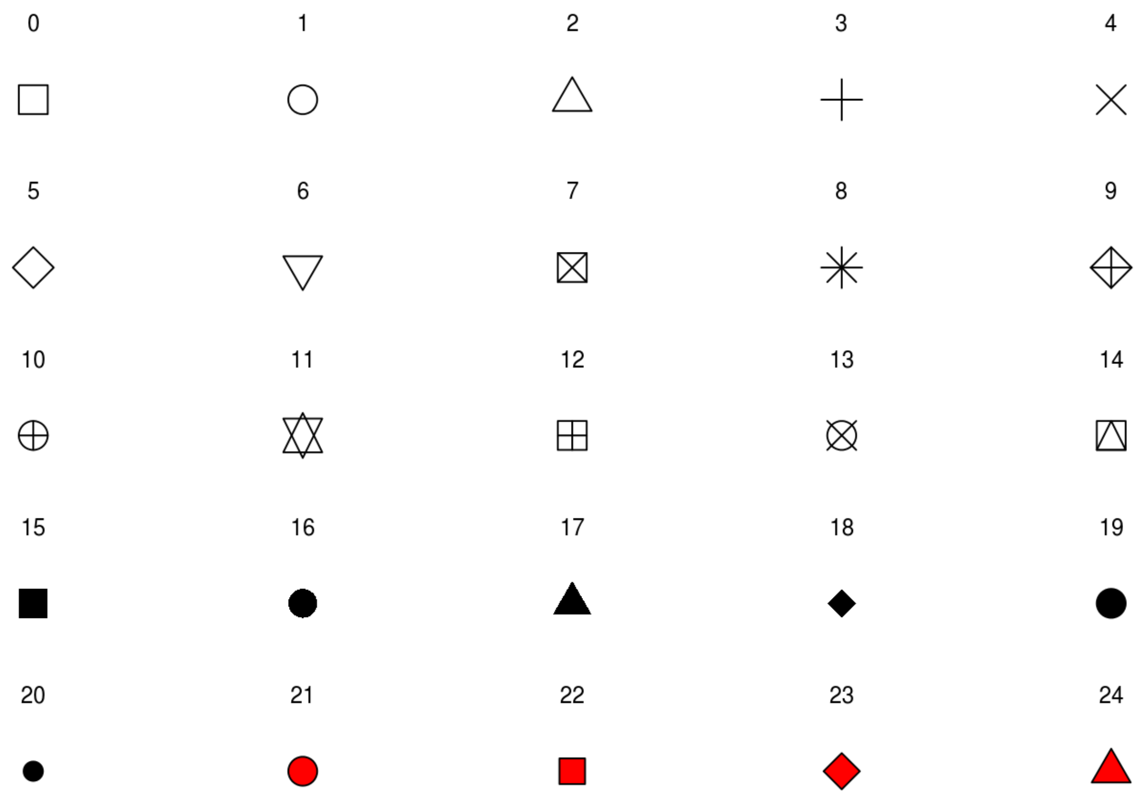
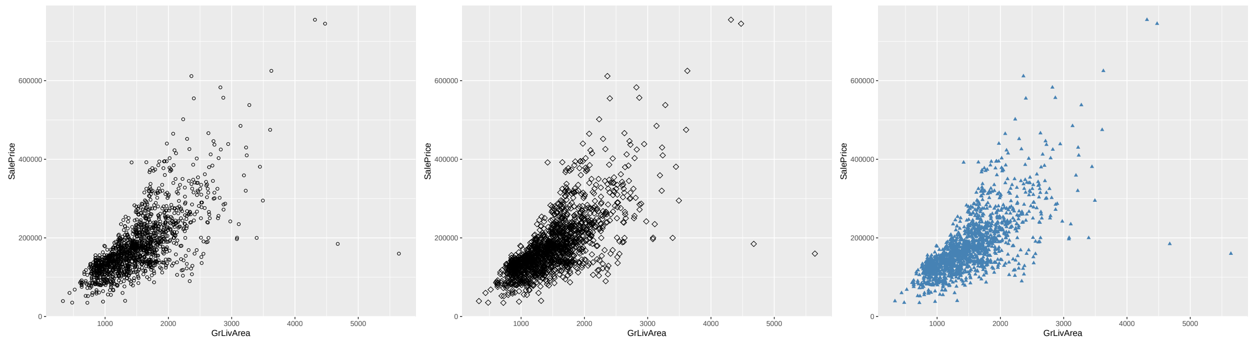


Figure 1: ...



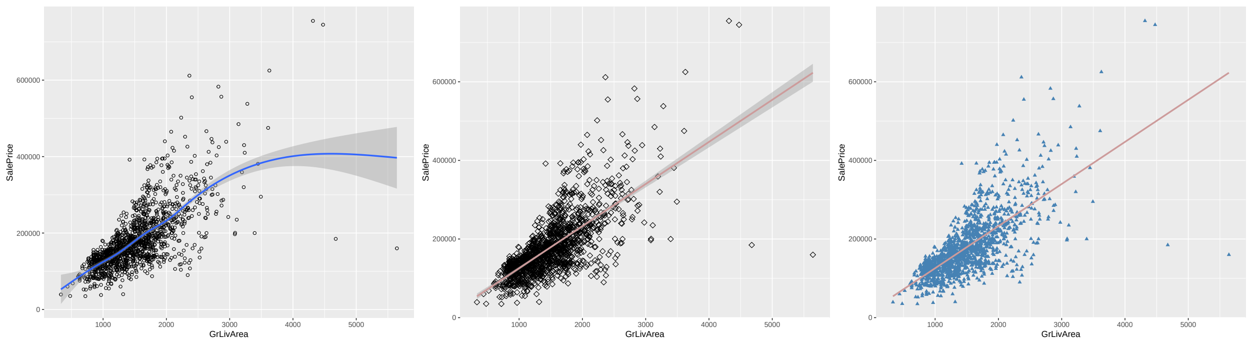
Some other available shapes are listed below:

Add smoothed conditional mean to the scatter plots

For detailed info check: https://ggplot2.tidyverse.org/reference/geom_smooth.html

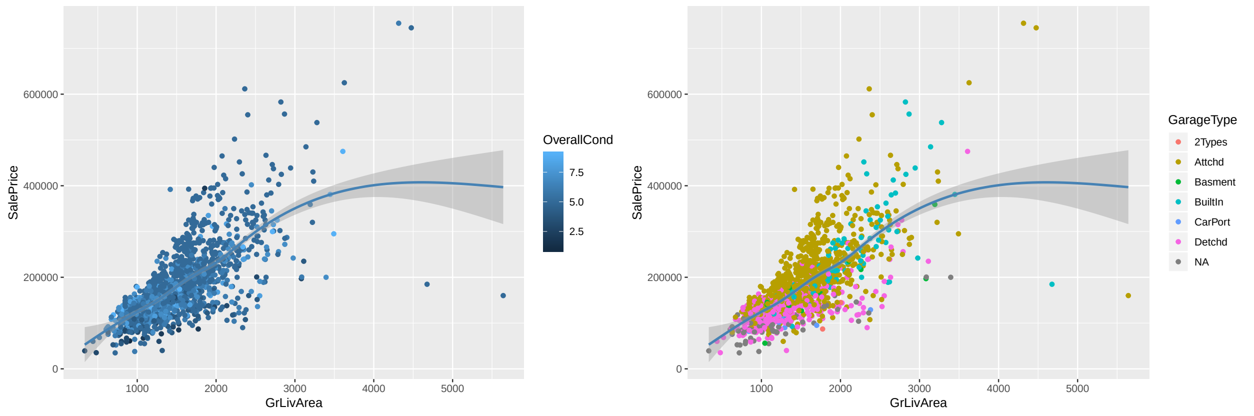
```
# Add regression lines to scatterplots
gg4 <- gg_init + geom_point(shape=1) + geom_smooth() # by default use method 'loess' and formula 'y ~ x
gg5 <- gg_init + geom_point(shape=5, size=2) +
                 geom_smooth(method=lm, colour='rosybrown3')
gg6 <- gg_init + geom_point(shape=17, size=1.5, colour='steelblue') +
                 geom_smooth(method=lm, colour='rosybrown3', se=FALSE) # Remove the shaded confidence r
```

```
grid.arrange(gg4, gg5, gg6, ncol=3)
```



What if we want to use different colors to reflect the value in another column?

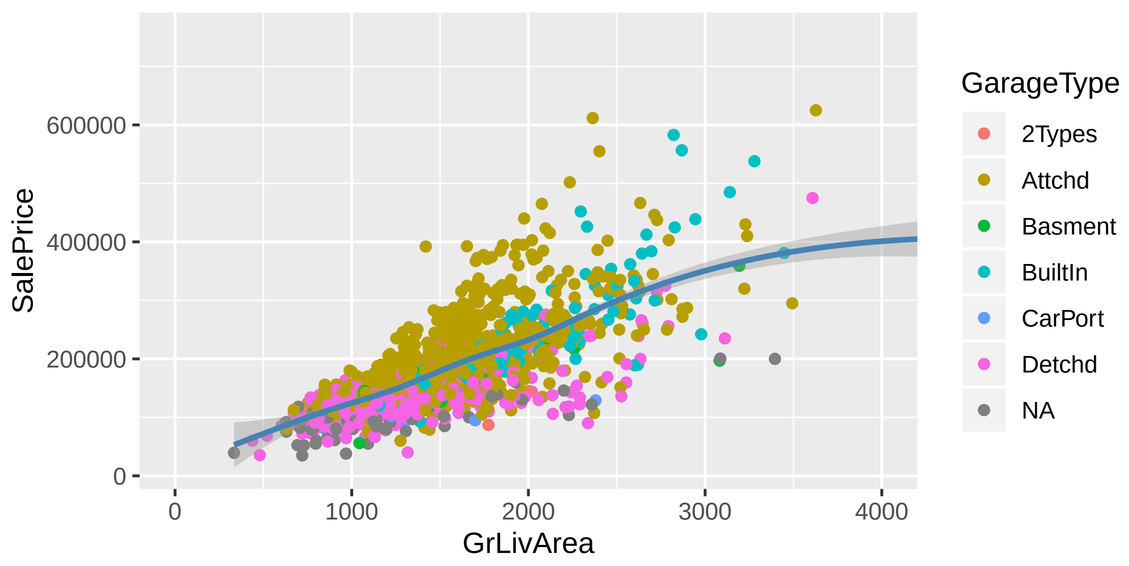
```
# Numerical(quantitative) value: OverallCond
gg7 <- gg_init + geom_point(aes(col=OverallCond)) + # Color the dots by OverallCond
                 geom_smooth(col="steelblue")

# Categorical(qualitative) value: Condition1
gg8 <- gg_init + geom_point(aes(col=GarageType)) + # Color the dots by SaleType
                 geom_smooth(col="steelblue") # change color palette

grid.arrange(gg7, gg8, ncol=2)
```



Adjust x and y axis limits

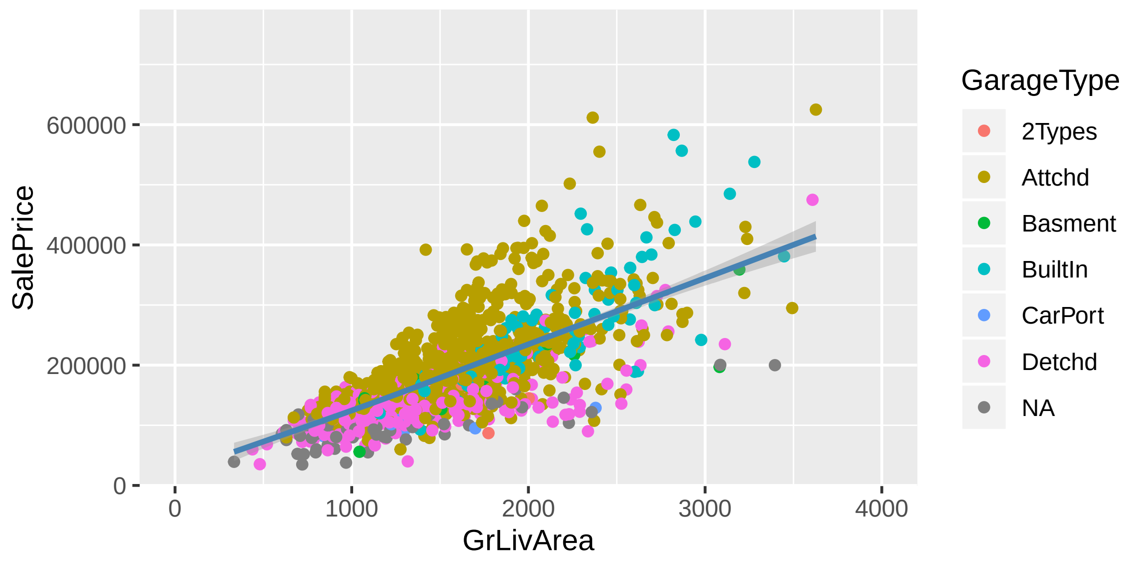
From the charts you can see that the fitted line is greatly affected by the points with GRLivArea > 4000, which seems to be outliers. Let's take a closer look at the data.

```
# zoom in to the region of interest without deleting the points
gg9 <- gg_init + geom_point(aes(col=GarageType)) + # Color the dots by SaleType
                 geom_smooth(col="steelblue") + # change color palette + coord_cartesian(xlim=c(0,
                 coord_cartesian(xlim=c(0, 4000))
print(gg9)
```



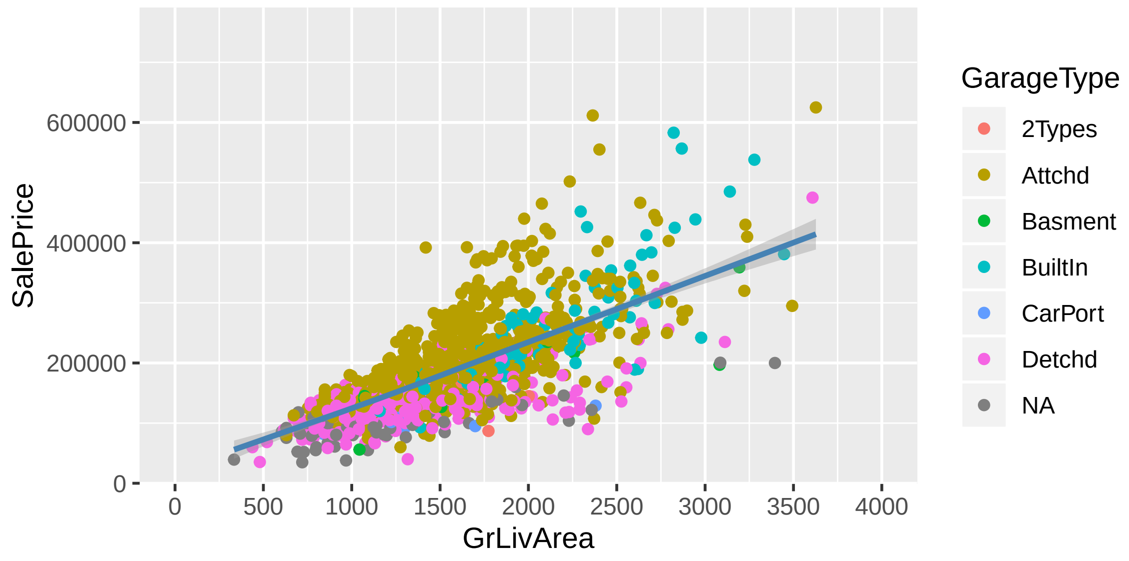
It seems to be safe to delete these outliers.

```r
# let's reset the xlim to delete the points outside the range
gg10 <- gg_init + geom_point(aes(col=GarageType)) + # Color the dots by SaleType
                  geom_smooth(col="steelblue") +
                  xlim(c(0, 4000))
print(gg10)
```

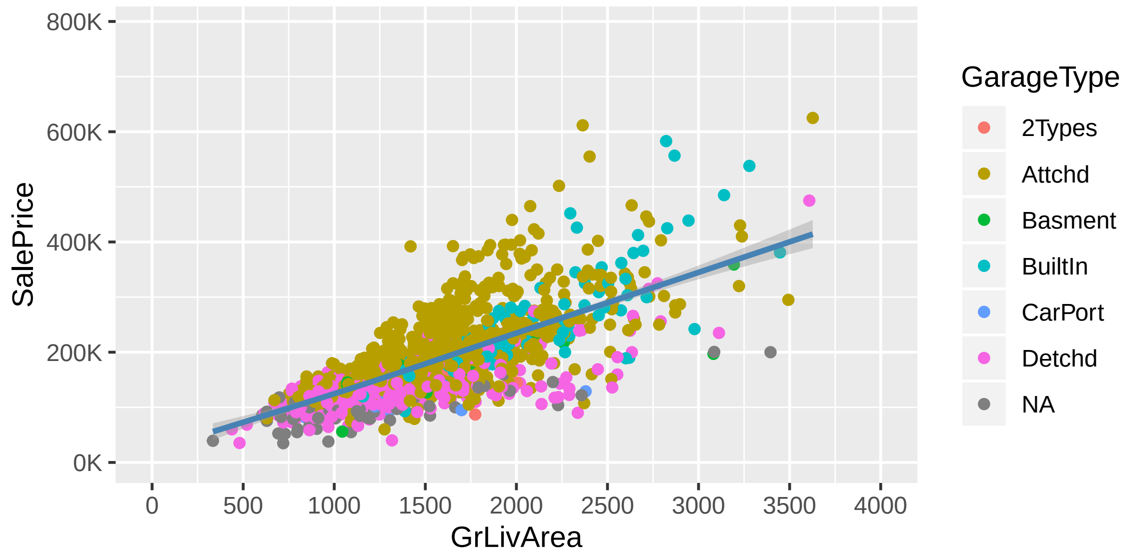


Customize tick marks and labels

```r
# Reset the breaks for x axis
gg11 <- gg_init + geom_point(aes(col=GarageType)) + # Color the dots by SaleType
                  geom_smooth(col="steelblue") +
                  scale_x_continuous(limits=c(0, 4000), breaks=seq(0, 4000, 500))

print(gg11)
```



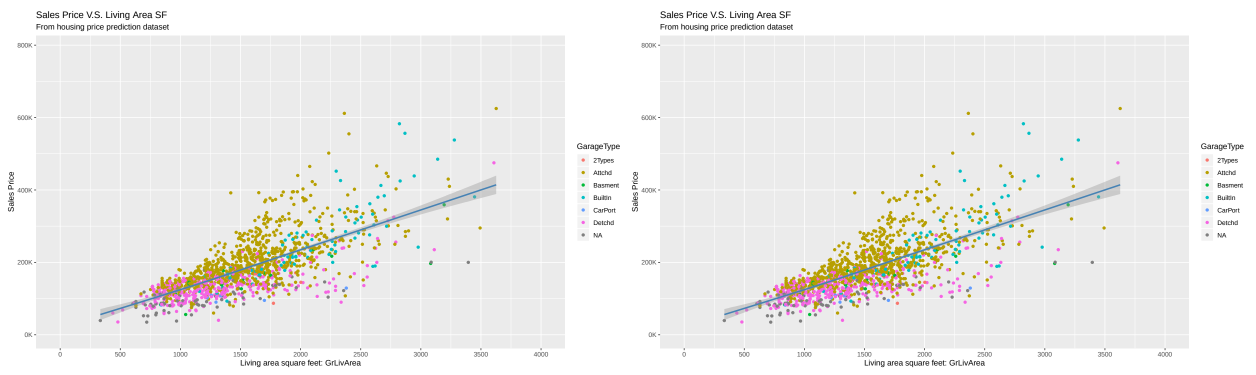
```
# Reset the tick marks for y axis
gg12 <- gg_init + geom_point(aes(col=GarageType)) + # Color the dots by SaleType
                  geom_smooth(col="steelblue") +
                  scale_x_continuous(limits=c(0, 4000), breaks=seq(0, 4000, 500)) +
                  scale_y_continuous(breaks=seq(0, 800000, 200000),
                                     labels = function(x){paste0(x/1000, 'K')},
                                     expand = c(0.1, 0.1))
print(gg12)
```



Add title and axis labels

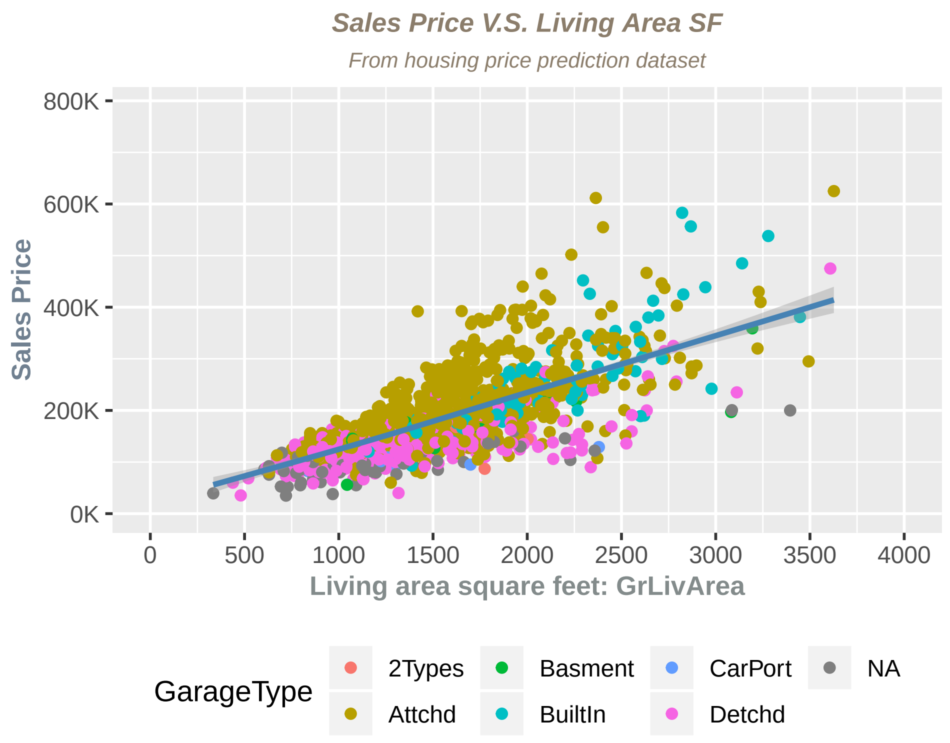
```
# Method 1
gg13 <- gg12 + labs(title = "Sales Price V.S. Living Area SF",
                    subtitle = "From housing price prediction dataset",
                    x = "Living area square feet: GrLivArea",
                    y = "Sales Price")
```

```
# Method 2
gg14 <- gg12 + ggtitle(label = 'Sales Price V.S. Living Area SF',
                          subtitle = "From housing price prediction dataset") +
                  xlab('Living area square feet: GrLivArea') +
                  ylab('Sales Price')

grid.arrange(gg13, gg14, ncol=2)
```



Adjust the size, color, font and position of title, legend and axis labels

```
gg15 <- gg14 + theme(
                      legend.position= "bottom", # Try: c(.5, .5), "None", "right", "top", "left"
                      plot.title = element_text(color="bisque4",
                                                size=10, face="bold.italic",
                                                hjust = 0.5),
                      plot.subtitle = element_text(color="bisque4",
                                                  size=8, face="italic",
                                                  hjust = 0.5),
                      axis.title.x = element_text(color="azure4", size=10, face="bold"),
                      axis.title.y = element_text(color="slategrey", size=10, face="bold")
                    )
print(gg15)
```

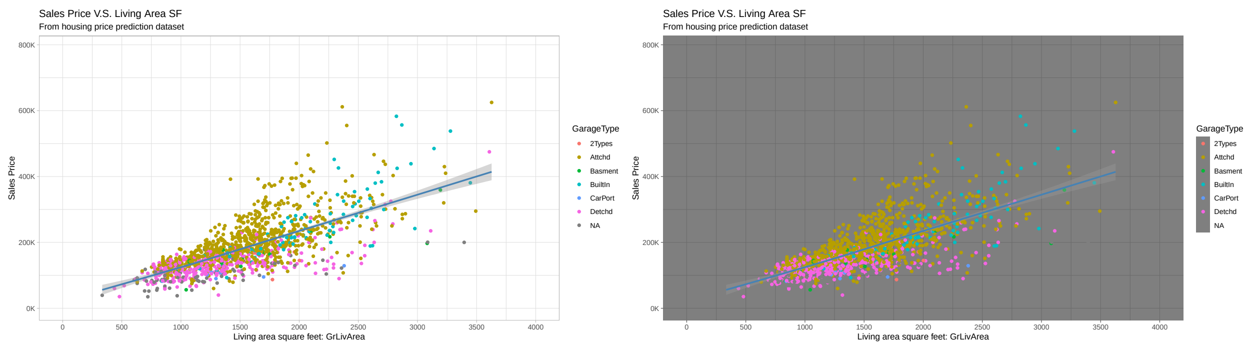


Customize the entire theme using ggthemes

You can easily change the theme without setting all the attibutes yourself using ggthemes. Reference: https://ggplot2.tidyverse.org/reference/ggtheme.html

Some of commonly used themes:

```r
# install.packages('ggthemes', dependencies = TRUE)
library(ggthemes)
theme_1 <- gg15 + theme_light()
theme_2 <- gg15 + theme_dark()
#theme_3 <- gg15 + theme_minimal()
#theme_4 <- gg15 + theme_gray()
#theme_5 <- gg15 + theme_bw()
#theme_6 <- gg15 + theme_void()
grid.arrange(theme_1, theme_2, ncol=2)
```



END