The ggplot Bible

Kevin da Silva Castanheira

August 14, 2018

# Getting Started

### Start by loading the Library dplyr, ggplot2, grid, and gridExtra

library(dplyr)  
library(ggplot2)  
library(grid)  
library(gridExtra)  
Quebec = read.csv("~/coding-club/advanced\_ggplot/Quebec.csv")

# ggplot & dplyr Basics

Let us refresh ourselves with the basics of dplyr and ggplot.

## dplyr

You can use dplyr to manipulate your data and make new columns. Here we will focus on summarizing data (i.e. data reduction).

Let’s take a look at the dataframe Quebec. It has 26 columns and a bunch of rows reflecting restaurants in the greater montreal area and their corresponding Yelp reviews.

head(Quebec, 2)

## business\_id name neighborhood  
## 1 O8S5hYJ1SMc8fA4QBtVujA La Bastringue Rosemont-La Petite-Patrie  
## 2 bBUMib8l6Me1ZB1\_Qkezkg Mysore Indian Cuisine Plateau-Mont-Royal  
## address city state postal\_code stars review\_count  
## 1 1335 rue Beaubien E MONTREAL QC H2G 4.0 -0.3431366  
## 2 4216 Saint-Laurent Boul MONTREAL QC H2W 3.5 -0.1624209  
## is\_open categories  
## 1 0 Breakfast & Brunch, Restaurants, French, Sandwiches, Cafes  
## 2 1 Restaurants, Pakistani, Indian, Middle Eastern  
## Restaurant Coffee Bar Brunch Total\_Restaurant Portuguese Italian  
## 1 TRUE TRUE FALSE TRUE -0.9839980 FALSE FALSE  
## 2 TRUE FALSE FALSE FALSE 0.4529971 FALSE FALSE  
## Chinese Mexican French Canadian Japanese American Indian PriceRange  
## 1 FALSE FALSE TRUE FALSE FALSE FALSE FALSE 2  
## 2 FALSE FALSE FALSE FALSE FALSE FALSE TRUE 2

summary(Quebec)

## business\_id name   
## \_\_S1Mhl6Lve4IbK3p4E19Q: 1 Frite Alors : 4   
## \_-u3js\_j6WHkRZ416gBeog: 1 Mikes : 4   
## \_cssId9VUaKqhMGxC4-5CQ: 1 3 Amigos : 3   
## \_Fy3ALeANfS\_ziaB7wOE8g: 1 Domino's Pizza : 3   
## \_K63HbZBVQSBCvQicQdl-A: 1 Les Enfants Terribles: 3   
## \_kPz9MKgbH1As9TIfpxTqA: 1 Portovino : 3   
## (Other) :1021 (Other) :1007   
## neighborhood  
## Ville-Marie :358   
## Plateau-Mont-Royal :196   
## Rosemont-La Petite-Patrie : 88   
## Laval : 62   
## Sud-Ouest : 52   
## Villeray-Saint-Michel-Parc-Extension: 37   
## (Other) :234   
## address city state   
## 1059 Rue Wellington : 2 MONTREAL :859 QC:1027   
## 1365 Rue Ontario E : 2 LAVAL : 61   
## 1437 Rue Crescent : 2 VERDUN : 24   
## 1448 Rue Saint Mathieu : 2 DOLLARD DES ORMEAUX: 13   
## 1812 Rue Sainte-Catherine O: 2 LASALLE : 11   
## 2137 Rue De Bleury : 2 ST LAURENT : 11   
## (Other) :1015 (Other) : 48   
## postal\_code stars review\_count is\_open   
## H2X : 68 Min. :1.000 Min. :-0.36895 Min. :0.0000   
## H2S : 63 1st Qu.:3.500 1st Qu.:-0.31732 1st Qu.:0.0000   
## H2Y : 62 Median :4.000 Median :-0.21405 Median :1.0000   
## H2Z : 56 Mean :3.678 Mean : 0.04059 Mean :0.7439   
## H3G : 52 3rd Qu.:4.000 3rd Qu.: 0.03120 3rd Qu.:1.0000   
## H2J : 46 Max. :5.000 Max. :13.75269 Max. :1.0000   
## (Other):680   
## categories Restaurant Coffee Bar   
## Restaurants, Italian: 46 Mode:logical Mode :logical Mode :logical   
## French, Restaurants : 45 TRUE:1027 FALSE:967 FALSE:923   
## Italian, Restaurants: 37 TRUE :60 TRUE :104   
## Restaurants, Chinese: 36   
## Restaurants, French : 36   
## Chinese, Restaurants: 34   
## (Other) :793   
## Brunch Total\_Restaurant Portuguese Italian   
## Mode :logical Min. :-1.46922 Mode :logical Mode :logical   
## FALSE:948 1st Qu.:-0.94667 FALSE:981 FALSE:816   
## TRUE :79 Median : 0.04243 TRUE :46 TRUE :211   
## Mean : 0.01871   
## 3rd Qu.: 0.58363   
## Max. : 1.85267   
##   
## Chinese Mexican French Canadian   
## Mode :logical Mode :logical Mode :logical Mode :logical   
## FALSE:881 FALSE:956 FALSE:808 FALSE:882   
## TRUE :146 TRUE :71 TRUE :219 TRUE :145   
##   
##   
##   
##   
## Japanese American Indian PriceRange   
## Mode :logical Mode :logical Mode :logical Min. :1.000   
## FALSE:901 FALSE:933 FALSE:954 1st Qu.:2.000   
## TRUE :126 TRUE :94 TRUE :73 Median :2.000   
## Mean :2.141   
## 3rd Qu.:2.000   
## Max. :4.000   
##

Now let’s try to get a mean rating for each postal code. The functions group\_by will split your data into groups, piping (%>%) that to the function summarise will produce a summary variable.

Quebec %>% group\_by(postal\_code) %>% summarise(Mean = mean(stars))

## # A tibble: 87 x 2  
## postal\_code Mean  
## <fct> <dbl>  
## 1 H1B 4   
## 2 H1E 4   
## 3 H1G 3.25  
## 4 H1L 3.33  
## 5 H1M 3   
## 6 H1N 3.38  
## 7 H1P 3.25  
## 8 H1S 3.33  
## 9 H1V 3.5   
## 10 H1W 3.92  
## # ... with 77 more rows

You can get mode than one outcome variable in summarise.

Quebec %>% group\_by(postal\_code) %>% summarise(Mean = mean(stars), SD= sd(stars), N = n())

## # A tibble: 87 x 4  
## postal\_code Mean SD N  
## <fct> <dbl> <dbl> <int>  
## 1 H1B 4 NA 1  
## 2 H1E 4 NA 1  
## 3 H1G 3.25 1.06 2  
## 4 H1L 3.33 0.764 3  
## 5 H1M 3 1.22 4  
## 6 H1N 3.38 0.618 13  
## 7 H1P 3.25 0.645 4  
## 8 H1S 3.33 0.816 6  
## 9 H1V 3.5 1.17 5  
## 10 H1W 3.92 0.886 13  
## # ... with 77 more rows

And you can choose more than one variable to group by.

Quebec %>% group\_by(postal\_code, PriceRange) %>% summarise(Mean = mean(stars), SD= sd(stars), N = n())

## # A tibble: 191 x 5  
## # Groups: postal\_code [?]  
## postal\_code PriceRange Mean SD N  
## <fct> <int> <dbl> <dbl> <int>  
## 1 H1B 2 4 NA 1  
## 2 H1E 3 4 NA 1  
## 3 H1G 1 4 NA 1  
## 4 H1G 2 2.5 NA 1  
## 5 H1L 2 3.33 0.764 3  
## 6 H1M 2 2 NA 1  
## 7 H1M 3 3.33 1.26 3  
## 8 H1N 1 3.5 NA 1  
## 9 H1N 2 3.39 0.651 9  
## 10 H1N 3 3.33 0.764 3  
## # ... with 181 more rows

## ggplot

ggplot works by taking your data and plotting it. I usually like to summarise my data and then pipe it to ggplot.

Every plot starts with the base ggplot statement. This is where you will tell R how you want your graph to be organized (what is your x, y, color variable etc.)

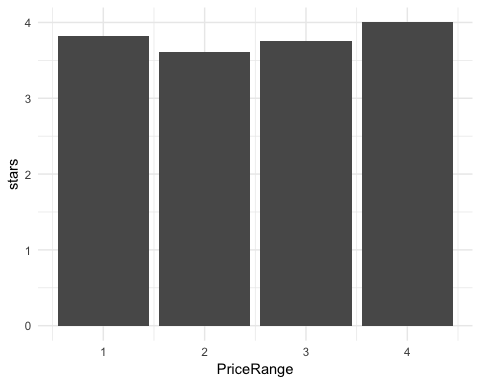
THIS ITSELF WONT PRODUCE A PLOT!!!!

You need to tell R what “shapes” you want it to draw. This is where geom comes in.

For example you can draw bar graphs:

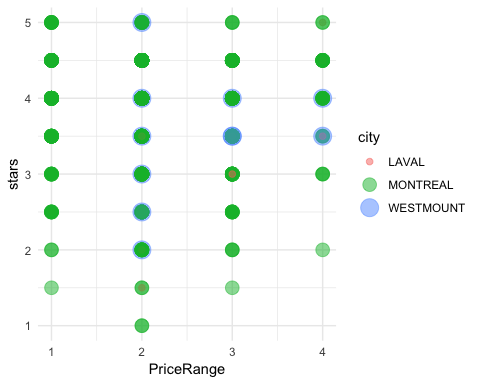
Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + theme\_minimal()

## No summary function supplied, defaulting to `mean\_se()



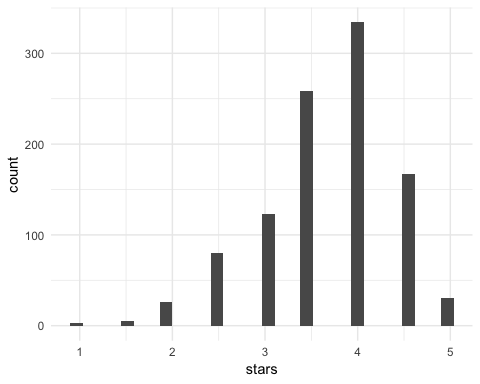
For example you can draw scatter plots:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city, size=city)) + geom\_point(alpha=0.5) + theme\_minimal()



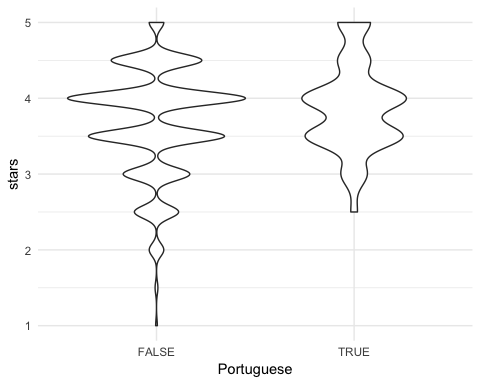
For example you can draw histograms:

Quebec %>%   
 ggplot(aes(stars)) + geom\_histogram() + theme\_minimal()



For example you can draw Violin plots:

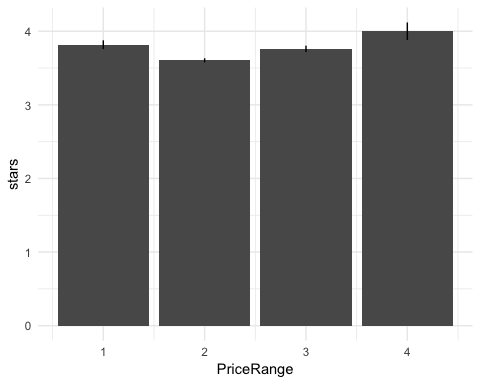
Quebec %>%   
 ggplot(aes(Portuguese, stars)) + geom\_violin() + theme\_minimal()

 … etc.

You can add geoms (error bars, best fit lines etc.) to these basic plots to create more complex plots.

Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal()

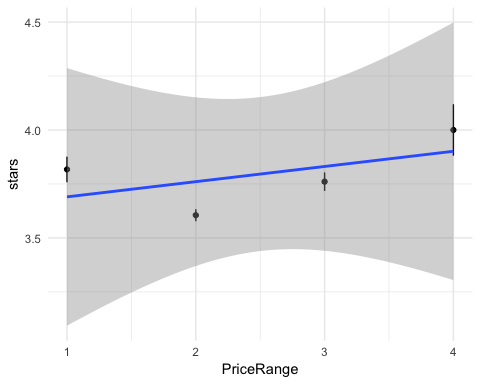
## No summary function supplied, defaulting to `mean\_se()



#### Best Fit lines

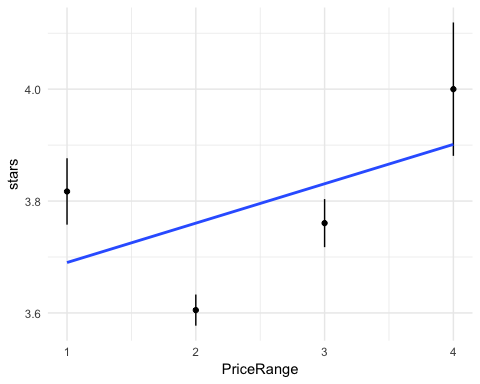
Best fit lines are added using the geom\_smooth() function. geom\_smooth() allows you to fit many kinds of fucntions but for now we will explore the linear fits.

Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_point() + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0) + geom\_smooth(method="lm") + theme\_minimal()



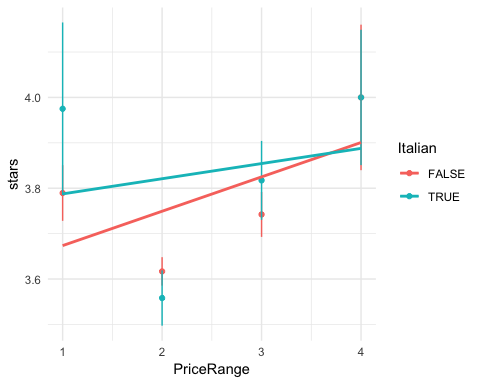
And you can remove the ugly shaded grey area (standard error) by supplying the function geom\_smooth with the argument se=FLASE.

Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_point() + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0) + geom\_smooth(method="lm", se=FALSE) + theme\_minimal()



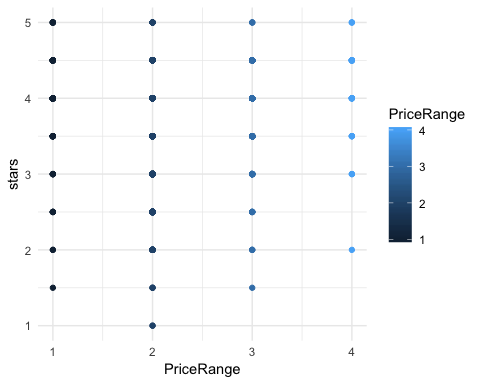
Specifying a color/fill/grouping variable will result in two lines fit–one for each group.

Quebec %>% group\_by(PriceRange, Italian) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars, color=Italian)) + geom\_point() + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0) + geom\_smooth(method="lm", se=FALSE) + theme\_minimal()

 ### Finishing touches #### Color, Size, alpha You can change the color, fill, size or alpha of your plot based on a variable in your dataset.

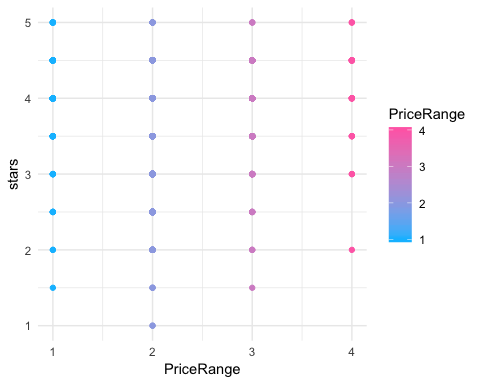
Setting color by a continuous (numeric) varibale will make a gradient

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() + theme\_minimal()

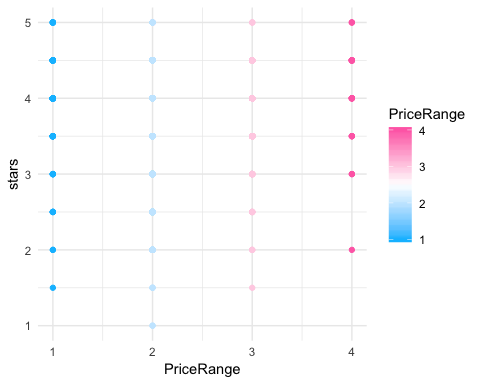


Which you can change:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() + scale\_color\_gradientn(colours=c("deepskyblue", "hotpink1")) + theme\_minimal()

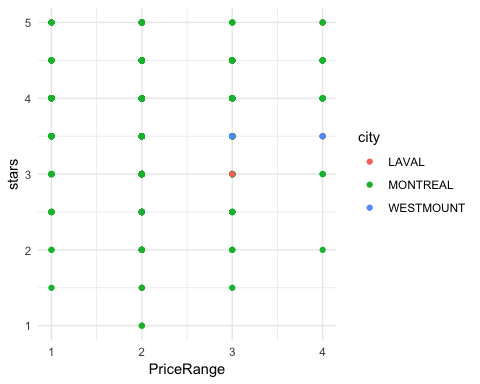


Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() + scale\_color\_gradientn(colours=c("deepskyblue", "white", "hotpink1")) + theme\_minimal()



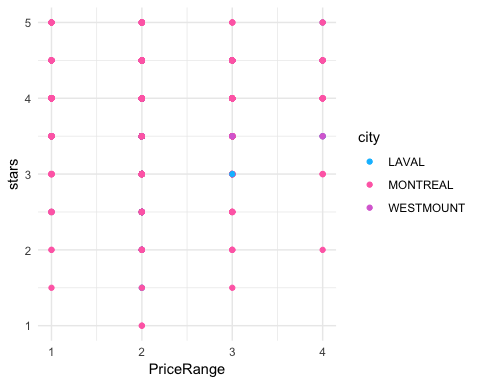
Or you can map color using discrete variables and not make a gradient:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city)) + geom\_point() + theme\_minimal()

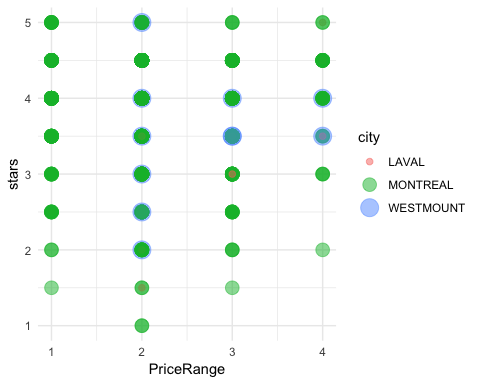


And you can change their values also:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city)) + geom\_point() + scale\_color\_manual(values=c("deepskyblue", "hotpink1", "orchid")) + theme\_minimal()

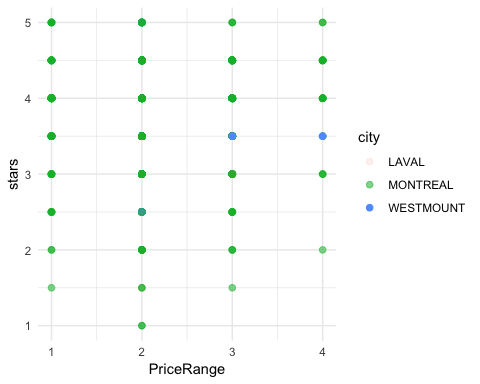
 For example here we have color and size mapping to city and opacity (alpha) set to 0.5:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city, size=city)) + geom\_point(alpha=0.5) + theme\_minimal()

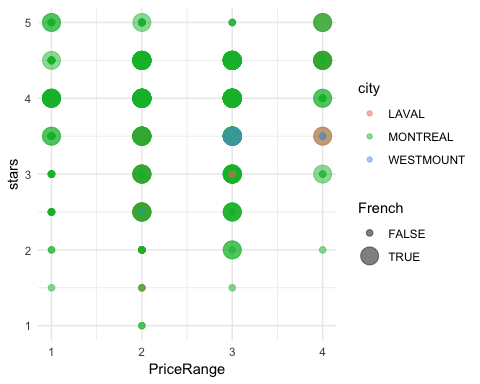


And here we have color and alpha mapping to city and size set to 2:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city, alpha=city)) + geom\_point(size=2) + theme\_minimal()

 You can even have color, alpha, size mapped to different variables:

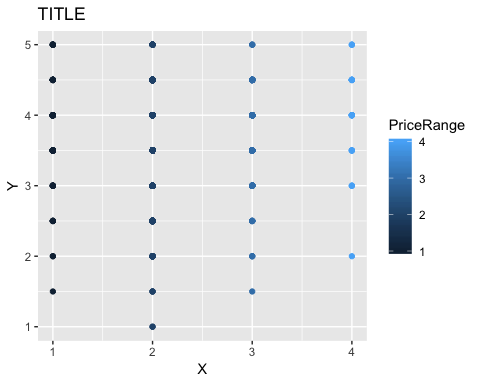
Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city, size=French)) + geom\_point(alpha=0.5) + theme\_minimal()



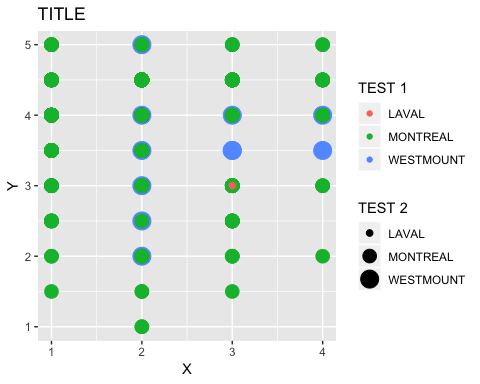
#### Title & Labels

You can title your graph by using the function ggtitle() and label you x and y axis with the functions xlab() and ylab()

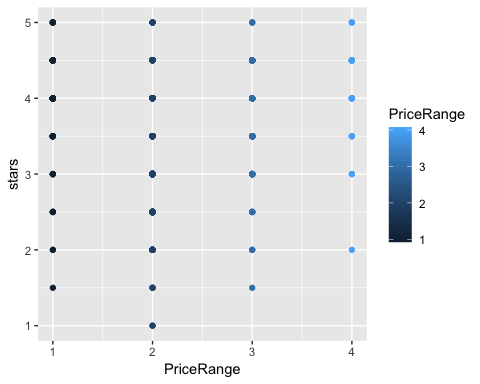
Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +ggtitle("TITLE") + xlab("X") + ylab("Y")

 You can even change the label of the legend:

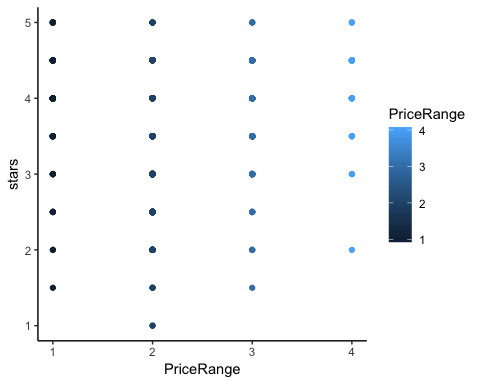
Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city, size=city)) + geom\_point() +ggtitle("TITLE") + xlab("X") + ylab("Y") + labs(color="TEST 1", size="TEST 2")

 #### Theme You can select a theme to change the look of your graph:

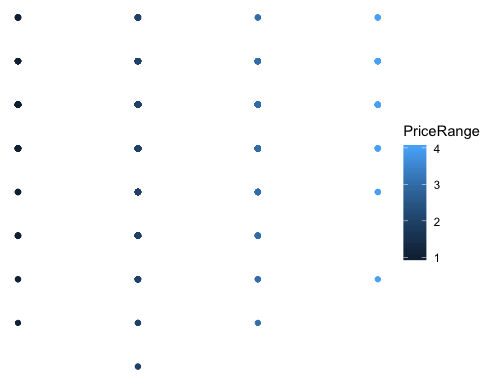
Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point()



Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() + theme\_classic()

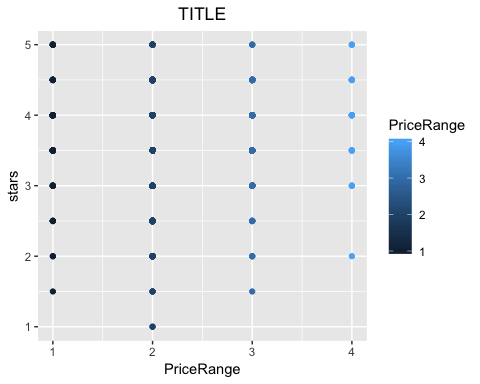


Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() + theme\_void()

 You can also manually change these values yourself to your liking including font size, centering the title, etc.

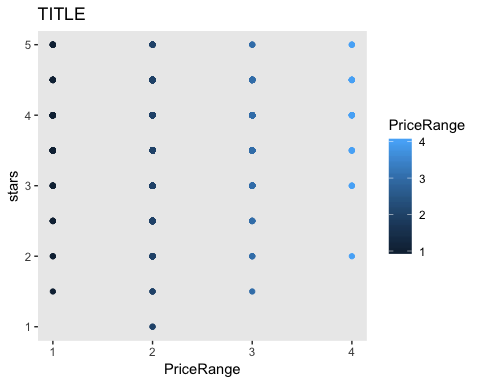
Center the title:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +  
 ggtitle("TITLE") + theme(plot.title = element\_text(hjust = 0.5))



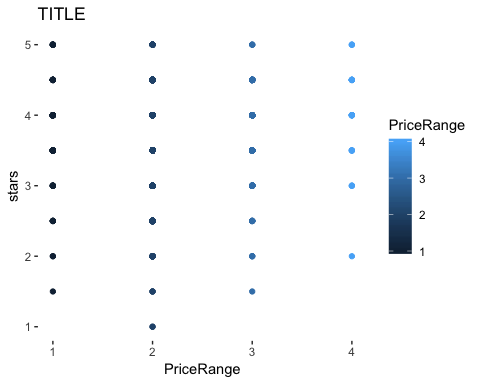
Remove the Grid lines:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +  
 ggtitle("TITLE") + theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank())



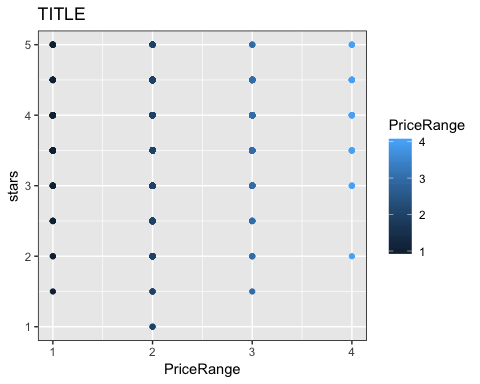
Remove the background color:

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +  
 ggtitle("TITLE") + theme(panel.background = element\_blank())



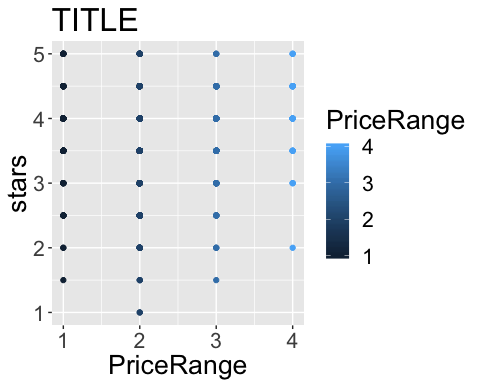
Change the border of the panels

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +  
 ggtitle("TITLE") + theme( panel.border = element\_rect(colour = "black", fill=NA, size=0.5))



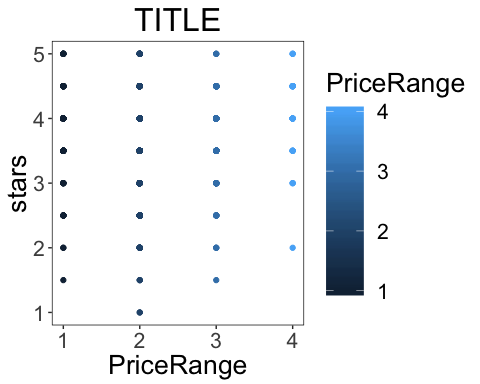
Or change the font size

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +  
 ggtitle("TITLE") + theme(text = element\_text(size=20))



Or do all of it together…

Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=PriceRange)) + geom\_point() +  
 ggtitle("TITLE") + theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(),  
 panel.background = element\_blank(), panel.border = element\_rect(colour = "black", fill=NA, size=0.5),  
 legend.key.size = unit(1,"cm"), text = element\_text(size=20),   
 plot.title = element\_text(hjust = 0.5))



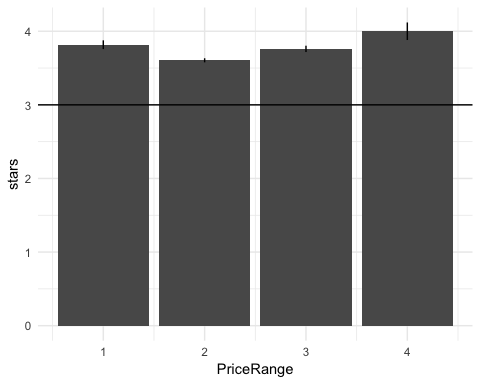
#### Horizontal and Verticle Lines

You can add lines onto your graphs to demark important values.

hline adds horizontal lines (i.e. to demark important y values like p=0.5 if plotting probabilities):

Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=3)

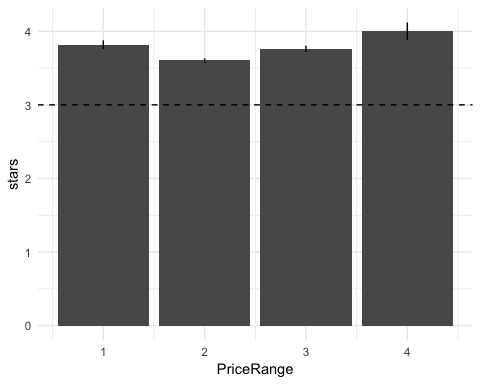
## No summary function supplied, defaulting to `mean\_se()



And you can change the style of the line by specifying what kind of line you want (or a number 1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash)

Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=3, linetype='dashed')

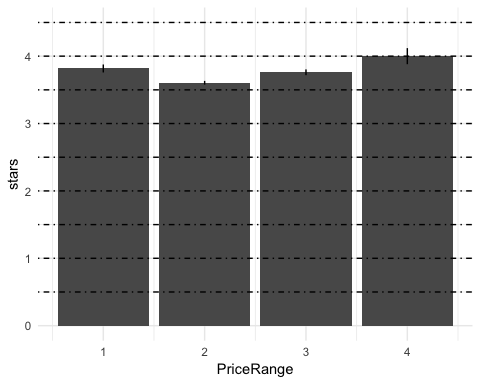
## No summary function supplied, defaulting to `mean\_se()



Or you can go crazy and add multiple!!!

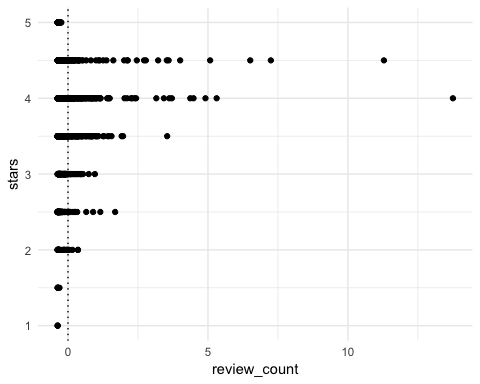
Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=seq(0.5,4.5, 0.5), linetype=4)

## No summary function supplied, defaulting to `mean\_se()

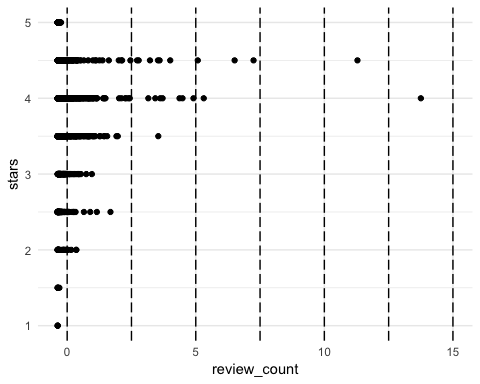


vline adds a verticle line (to demark important x values like stimulus onset):

Quebec %>%   
 ggplot(aes(review\_count, stars)) + geom\_point() + theme\_minimal() + geom\_vline(xintercept=0, linetype=3)



Quebec %>%   
 ggplot(aes(review\_count, stars)) + geom\_point() + theme\_minimal() + geom\_vline(xintercept=seq(0, 15, 2.5), linetype=5)

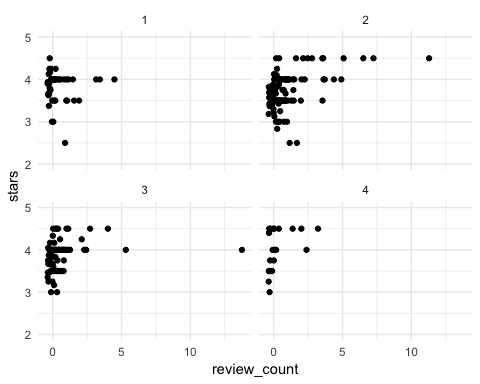


# Facet Wrapping

Facet wrapping is a way of separating your graph into panels based on variables within your dataset.

Quebec %>%  
 ggplot(aes(review\_count, stars)) + geom\_point(stat = "summary", position="dodge") + theme\_minimal() + facet\_wrap(~PriceRange)

## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()



You can specify how you want these plots to be arranged within your graph by specifying the number of columns or rows.

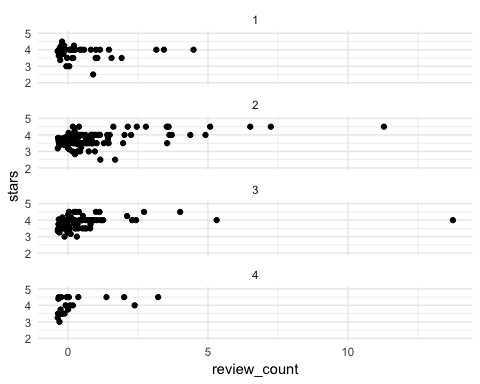
Quebec %>%  
 ggplot(aes(review\_count, stars)) + geom\_point(stat = "summary", position="dodge") + theme\_minimal() + facet\_wrap(~PriceRange, nrow=1)

## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()



Quebec %>%   
 ggplot(aes(review\_count, stars)) + geom\_point(stat = "summary", position="dodge") + theme\_minimal() + facet\_wrap(~PriceRange, ncol=1)

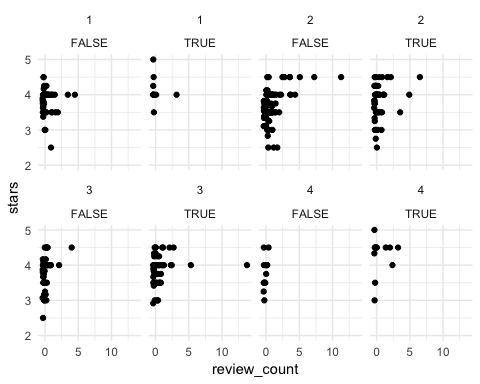
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()



You can facet wrap by multiple variables. The order of the variables you specify to facet wrap will change how the plot is made

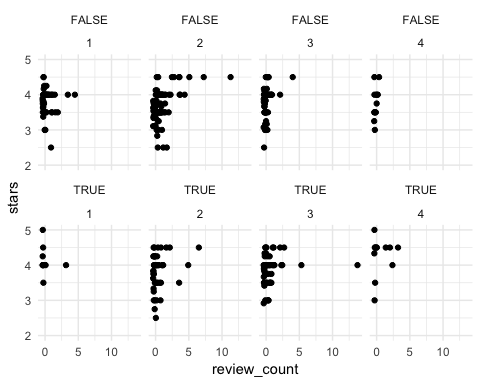
Quebec %>%   
 ggplot(aes(review\_count, stars)) + geom\_point(stat = "summary", position="dodge") + theme\_minimal() + facet\_wrap(~PriceRange+French, nrow=2, ncol=4)

## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()



Quebec %>%   
 ggplot(aes(review\_count, stars)) + geom\_point(stat = "summary", position="dodge") + theme\_minimal() + facet\_wrap(~French+PriceRange, nrow=2, ncol=4)

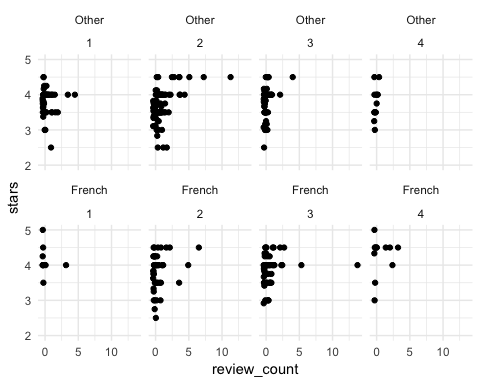
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()



You can pipe your data into functions like revalue form the plyr package (DO NOT LOAD THE PLYR PACKAGE AND DPLYR SIMULTANEOUSLY!!!!) to relabel your variables before graphing.

Quebec$French = plyr::revalue(as.factor(Quebec$French), c("TRUE"="French", "FALSE"="Other"))  
Quebec %>%  
 ggplot(aes(review\_count, stars)) + geom\_point(stat = "summary", position="dodge") + theme\_minimal() + facet\_wrap(~French+PriceRange, nrow=2, ncol=4)

## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()  
## No summary function supplied, defaulting to `mean\_se()



Facet Wrapping is very useful when you want to arrange multiple plots with the same dependant and independant variables.

But what happens when we want to plot multiple graphs in the same figure with different variables?

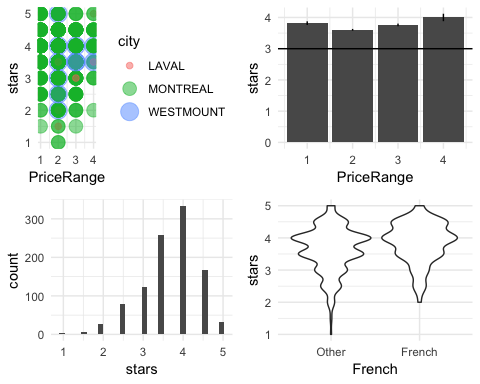
# Arranging Plots

With the libraries grid and gridExtra, you can arrange plots of different variables all within the same figure

plot1 <- Quebec %>% filter(city %in% c("MONTREAL", "LAVAL", "WESTMOUNT")) %>%  
 ggplot(aes(PriceRange, stars, color=city, size=city)) + geom\_point(alpha=0.5) + theme\_minimal()  
  
plot2 <- Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=3)  
  
plot3 <- Quebec %>%   
 ggplot(aes(stars)) + geom\_histogram() + theme\_minimal()  
  
plot4 <- Quebec %>%   
 ggplot(aes(French, stars)) + geom\_violin() + theme\_minimal()  
  
grid.arrange(plot1, plot2, plot3, plot4)

## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

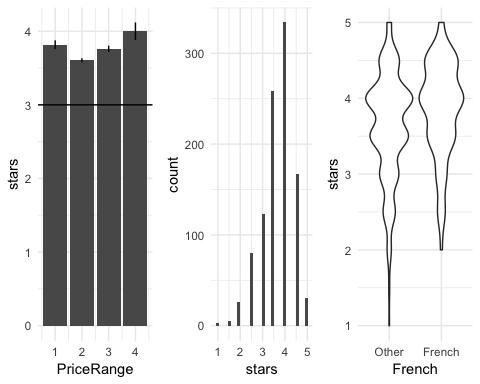


You can change the way the plots are arranged by specifying the number of rows and columns

grid.arrange(plot2, plot3, plot4, nrow=1)

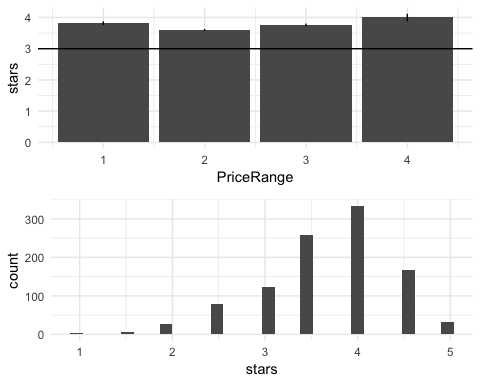
## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



grid.arrange(plot2, plot3, ncol=1, nrow=2)

## No summary function supplied, defaulting to `mean\_se()  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



You can achieve more complex designs (non-grid layouts) by specifying the widths and layouts you want.

Let’s explore our options.

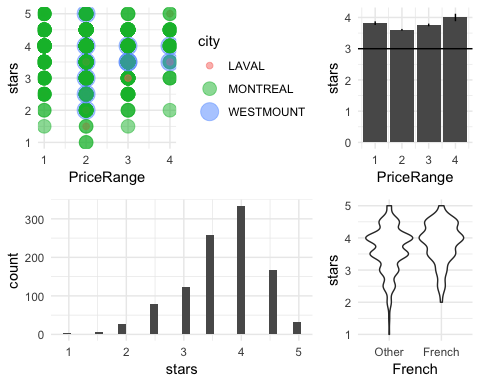
Let’s make a 2x2 figure where the left graphs are twice as wide as the right hand graphs. Width specifies the width of each column. Layout\_matrix specifies how the graphs are to be laid out, row-by-row.

grid.arrange(plot1, plot2, plot3, plot4,  
 widths = c(2, 1),  
 layout\_matrix = rbind(c(1, 2),  
 c(3, 4)))

## Warning: Using size for a discrete variable is not advised.

## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



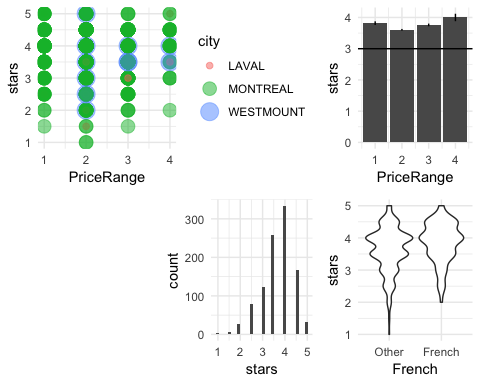
Now Let’s make only the top-left graph large and the others small. Notice how we have to leave a blank space on the bottom row for the graph that is missing and the first graph takes up two columns.

grid.arrange(plot1, plot2, plot3, plot4,  
 widths = c(1, 1, 1),  
 layout\_matrix = rbind(c(1, 1, 2),  
 c(NA, 3, 4)))

## Warning: Using size for a discrete variable is not advised.

## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



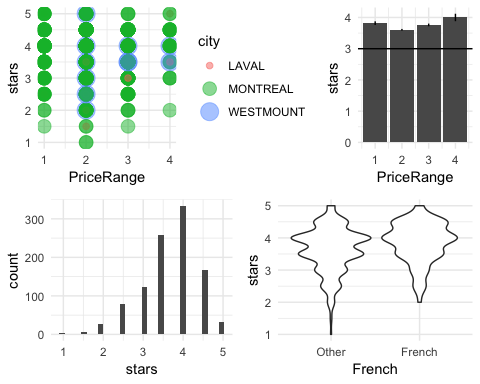
Now try making the top left graph be 3 times as large as the other graph in it’s row but have the bottom two split evenly. FOr this we’re going to need 6 columns.

grid.arrange(plot1, plot2, plot3, plot4,  
 widths = c(1, 1, 1, 1, 1, 1),  
 layout\_matrix = rbind(c(1, 1, 1, 1, 2, 2),  
 c(3, 3, 3, 4, 4, 4)))

## Warning: Using size for a discrete variable is not advised.

## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



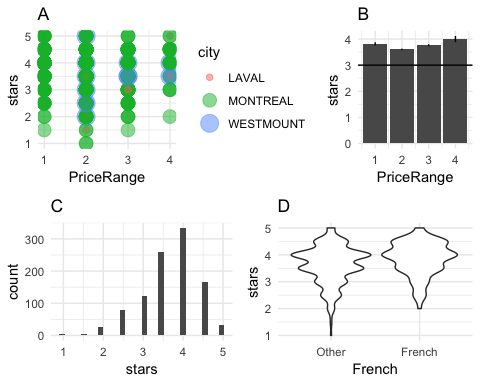
Now let’s add a titles to the subplots!

grid.arrange(plot1 + ggtitle("A"), plot2 + ggtitle("B"), plot3 + ggtitle("C"), plot4 + ggtitle("D"),  
 widths = c(1, 1, 1, 1, 1, 1),  
 layout\_matrix = rbind(c(1, 1, 1, 1, 2, 2),  
 c(3, 3, 3, 4, 4, 4)))

## Warning: Using size for a discrete variable is not advised.

## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



#### Titles

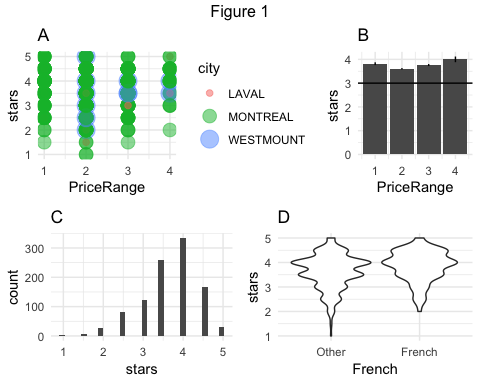
What about an general title.

grid.arrange(plot1 + ggtitle("A"), plot2 + ggtitle("B"), plot3 + ggtitle("C"), plot4 + ggtitle("D"),  
top=textGrob("Figure 1"),  
 widths = c(1, 1, 1, 1, 1, 1),  
 layout\_matrix = rbind(c(1, 1, 1, 1, 2, 2),  
 c(3, 3, 3, 4, 4, 4)))

## Warning: Using size for a discrete variable is not advised.

## No summary function supplied, defaulting to `mean\_se()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



#### Graph Inserts

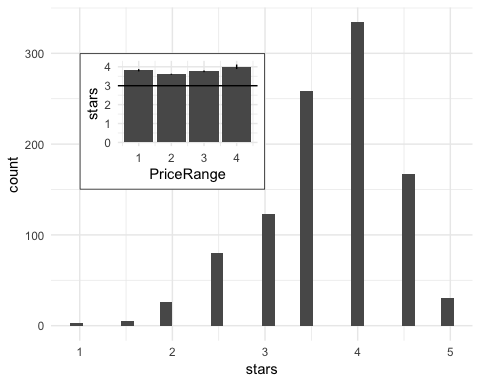
To insert a smaller graph in your other graph you first must turn your ggplot graph into a ggplorGrob. This will allow you to insert it into any figure.

plot2 <- Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=3)  
  
plot2Grob = ggplotGrob(plot2 + theme(plot.background = element\_rect(colour = "black")))

## No summary function supplied, defaulting to `mean\_se()

Quebec %>%   
 ggplot(aes(stars)) + geom\_histogram() + theme\_minimal() +  
 annotation\_custom(  
 grob = plot2Grob,  
 xmin = 1,  
 xmax = 3,  
 ymin = 150,  
 ymax = 300)

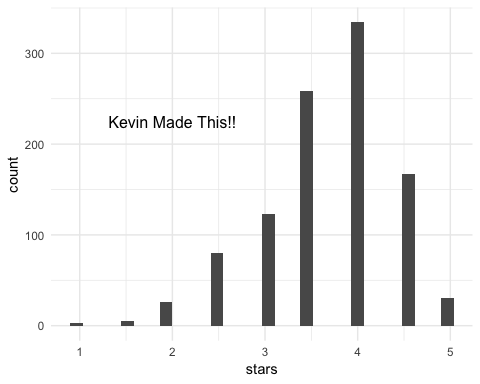
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



You can insert text.

plot2 <- Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=3)  
  
Grob\_insert = textGrob("Kevin Made This!!")  
  
Quebec %>%   
 ggplot(aes(stars)) + geom\_histogram() + theme\_minimal() +  
 annotation\_custom(  
 grob = Grob\_insert,  
 xmin = 1,  
 xmax = 3,  
 ymin = 150,  
 ymax = 300)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Or you can add a table which summarizes the average rating of Portuguese and non-Portuguese Restaurants.

plot2 <- Quebec %>% group\_by(PriceRange) %>% summarise(sem= sd(stars)/sqrt(length(stars)), stars=mean(stars)) %>%   
 ggplot(aes(PriceRange, stars)) + geom\_bar(stat = "summary", position="dodge") + geom\_errorbar(aes(ymin=stars-sem, ymax=stars+sem), width=0, position="dodge") + theme\_minimal() + geom\_hline(yintercept=3)  
  
Grob\_insert = tableGrob(Quebec %>% group\_by(Portuguese) %>% summarise(mean=mean(stars)), rows = c())  
  
Quebec %>%   
 ggplot(aes(stars)) + geom\_histogram() + theme\_minimal() +  
 annotation\_custom(  
 grob = Grob\_insert,  
 xmin = 1,  
 xmax = 3,  
 ymin = 150,  
 ymax = 300)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

