LA2 group 5

Dheeraj Reddy, Jatin Thakur

2023-05-14

LA2

load the libraries needed

```
library(readr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v purrr     1.0.1
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(gcookbook)
library(dplyr)
```

now load the data set.

```
sdata <- read.csv("C:\\Users\\Dheeraj's Vivobook\\Documents\\EDA LA\\supermarket_sales.csv")
head(sdata, 5)
```

```
##    Invoice.ID Branch      City Customer.type Gender           Product.line
## 1 750-67-8428      A    Yangon        Member Female      Health and beauty
## 2 226-31-3081      C Naypyitaw        Normal Female Electronic accessories
## 3 631-41-3108      A    Yangon        Normal   Male     Home and lifestyle
## 4 123-19-1176      A    Yangon        Member   Male      Health and beauty
## 5 373-73-7910      A    Yangon        Normal   Male      Sports and travel
##   Unit.price Quantity  Tax.5.    Total      Date  Time     Payment   cogs
## 1      74.69        7 26.1415 548.9715  1/5/2019 13:08     Ewallet 522.83
## 2      15.28        5  3.8200  80.2200  3/8/2019 10:29        Cash  76.40
## 3      46.33        7 16.2155 340.5255  3/3/2019 13:23 Credit card 324.31
## 4      58.22        8 23.2880 489.0480 1/27/2019 20:33     Ewallet 465.76
## 5      86.31        7 30.2085 634.3785  2/8/2019 10:37     Ewallet 604.17
##   gross.margin.percentage gross.income Rating
## 1                4.761905      26.1415    9.1
## 2                4.761905       3.8200    9.6
```

```
## 3               4.761905      16.2155    7.4
## 4               4.761905      23.2880    8.4
## 5               4.761905      30.2085    5.3
```

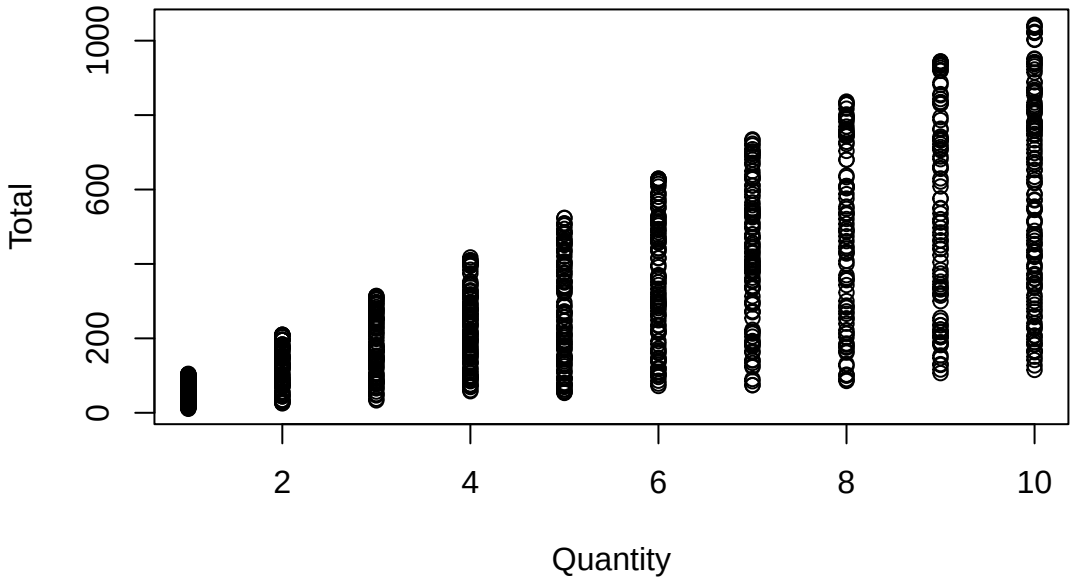
The structure of the data set is as follows:

```
str(sdata)
```

```
## 'data.frame':    1000 obs. of  17 variables:
##  $ Invoice.ID            : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176" ...
##  $ Branch                : chr  "A" "C" "A" "A" ...
##  $ City                  : chr  "Yangon" "Naypyitaw" "Yangon" "Yangon" ...
##  $ Customer.type         : chr  "Member" "Normal" "Normal" "Member" ...
##  $ Gender                : chr  "Female" "Female" "Male" "Male" ...
##  $ Product.line          : chr  "Health and beauty" "Electronic accessories" "Home and lifestyle" "
##  $ Unit.price            : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ Quantity              : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ Tax.5.                : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ Total                 : num  549 80.2 340.5 489 634.4 ...
##  $ Date                  : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Time                  : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ Payment               : chr  "Ewallet" "Cash" "Credit card" "Ewallet" ...
##  $ cogs                  : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross.margin.percentage: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ gross.income          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ Rating                : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
```

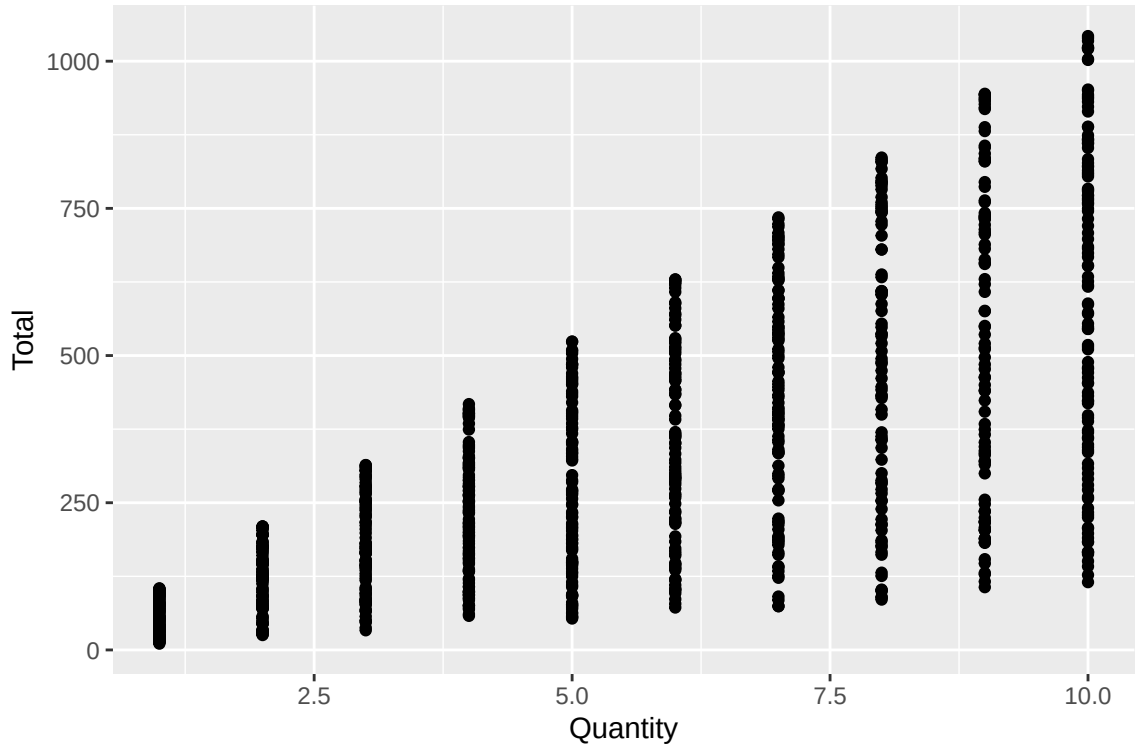
Below is an example to create a scatter plot between "Quantity" and "Total".

```
plot(sdata$Quantity, sdata$Total, xlab = "Quantity", ylab = "Total")
```



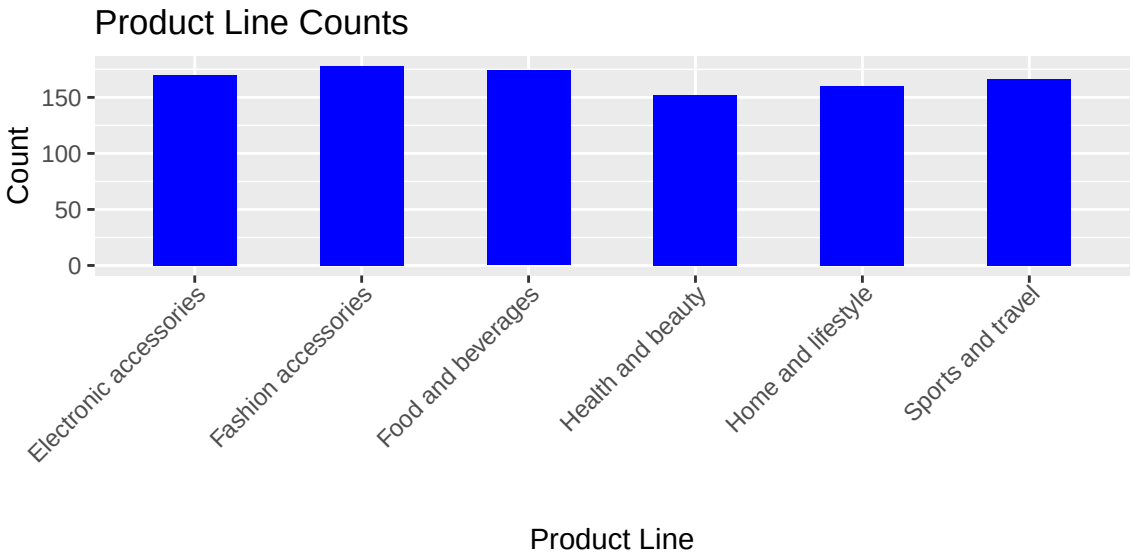
The same can be done using ggplot2.

```
library(ggplot2)
ggplot(sdata, aes(x=Quantity, y=Total)) + geom_point()
```



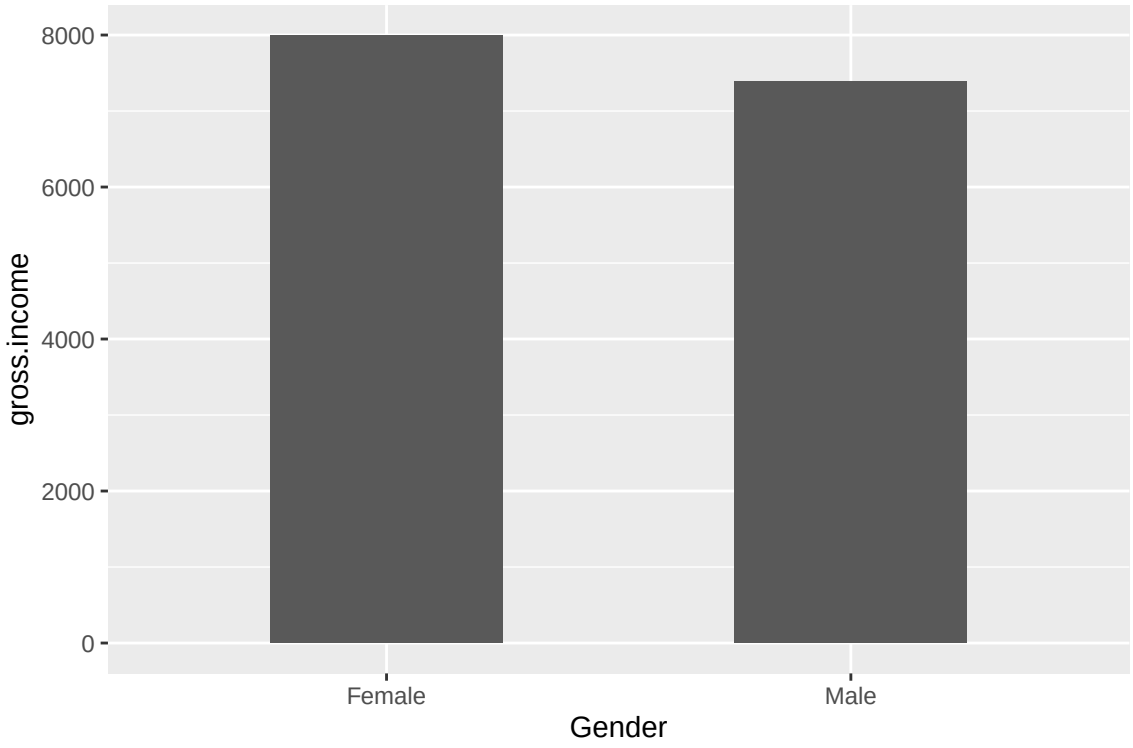
A bar plot can be created using ggplot.

```
ggplot(sdata, aes(x = Product.line)) +
geom_bar(fill="blue", width = 0.5) +
labs(x = "Product Line", y = "Count", title = "Product Line Counts") +
theme(axis.text.x = element_text(angle = 45, hjust = 1,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



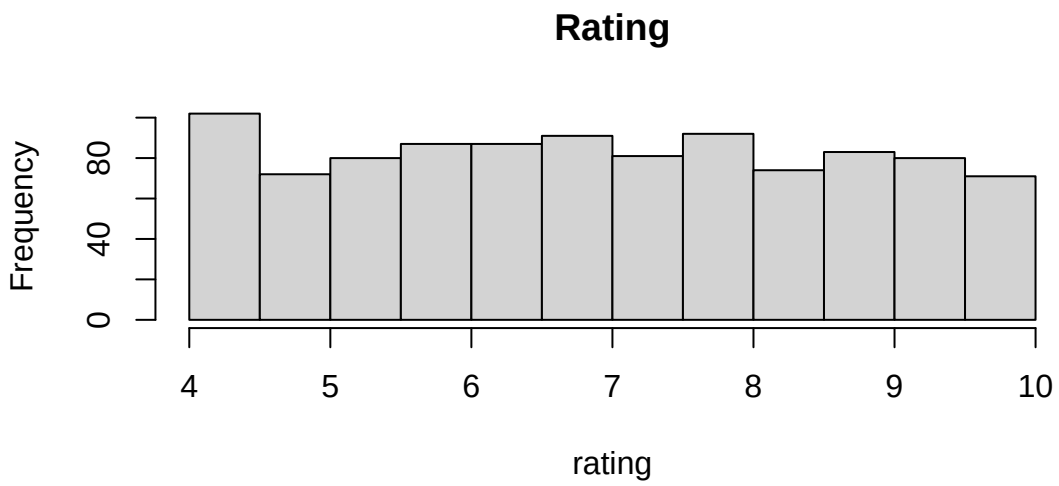
We can also use geom_col().

```
ggplot(sdata, aes(x=Gender, y=gross.income)) + geom_col(width = 0.5)
```



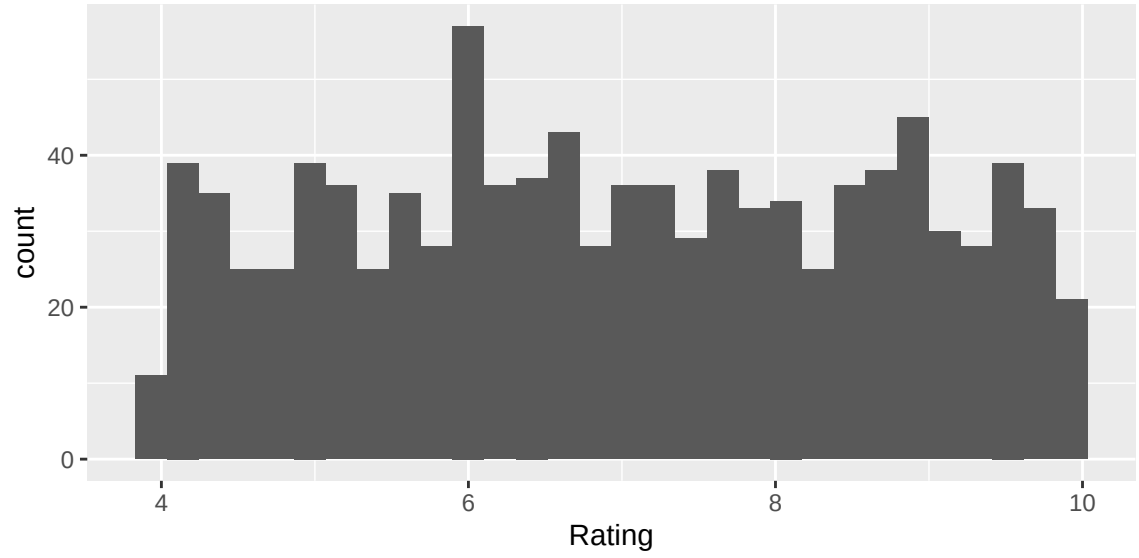
To create a histogram for, say Rating column, the below command will help.

```
hist(sdata$Rating, main = "Rating", xlab = "rating")
```



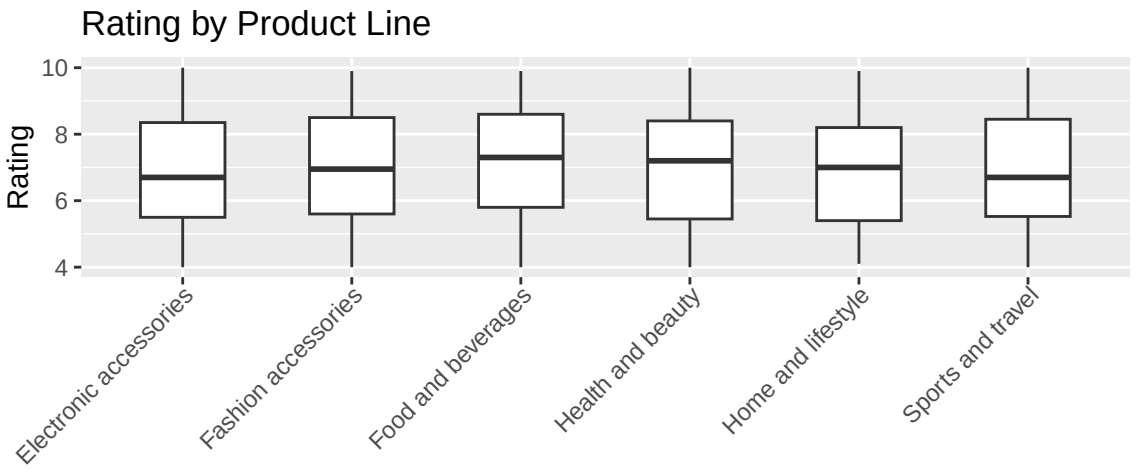

```
ggplot(sdata, aes(x = Rating)) + geom_histogram()
```

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



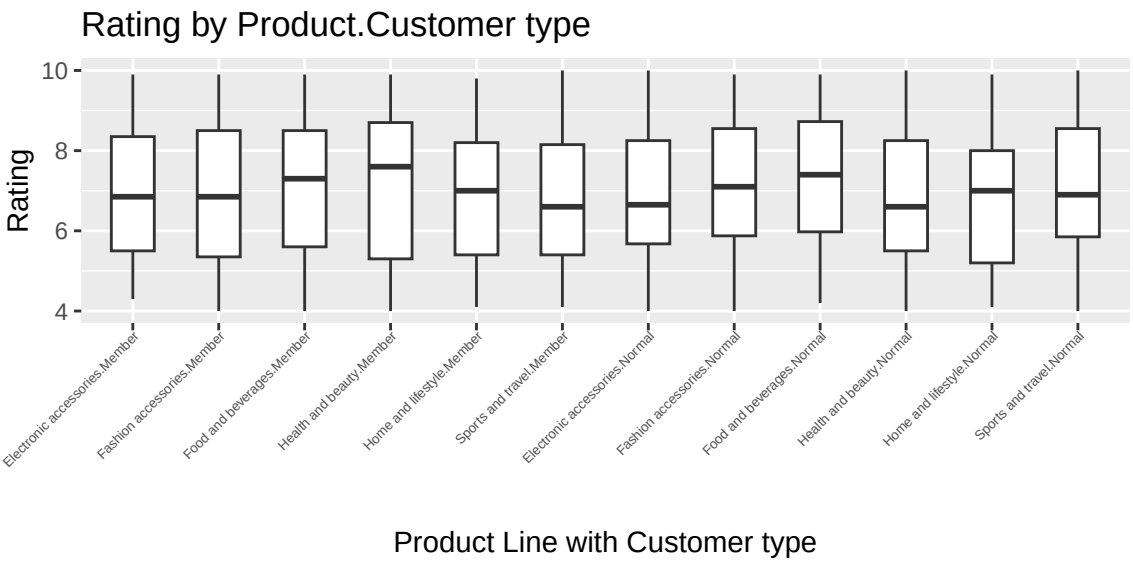
To create a box plot, we use geom_boxplot() using ggplot().

```
ggplot(sdata, aes(x = Product.line, y = Rating)) +
    geom_boxplot(width=0.5) +
    labs(x = "Product Line", y = "Rating", title = "Rating by Product Line") +                theme(ax
    margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



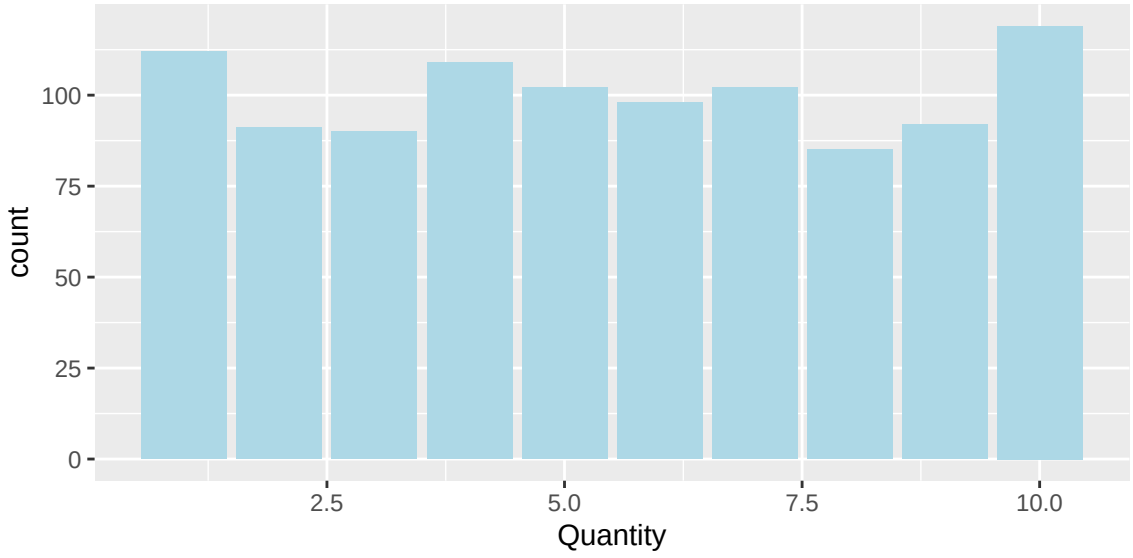
We can also use interaction() to combine variables.

```
ggplot(sdata, aes(x = interaction(Product.line, Customer.type), y = Rating)) + geom_boxplot(width=0.5) +
labs(x = "Product Line with Customer type", y = "Rating",
title = "Rating by Product.Customer type") +
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



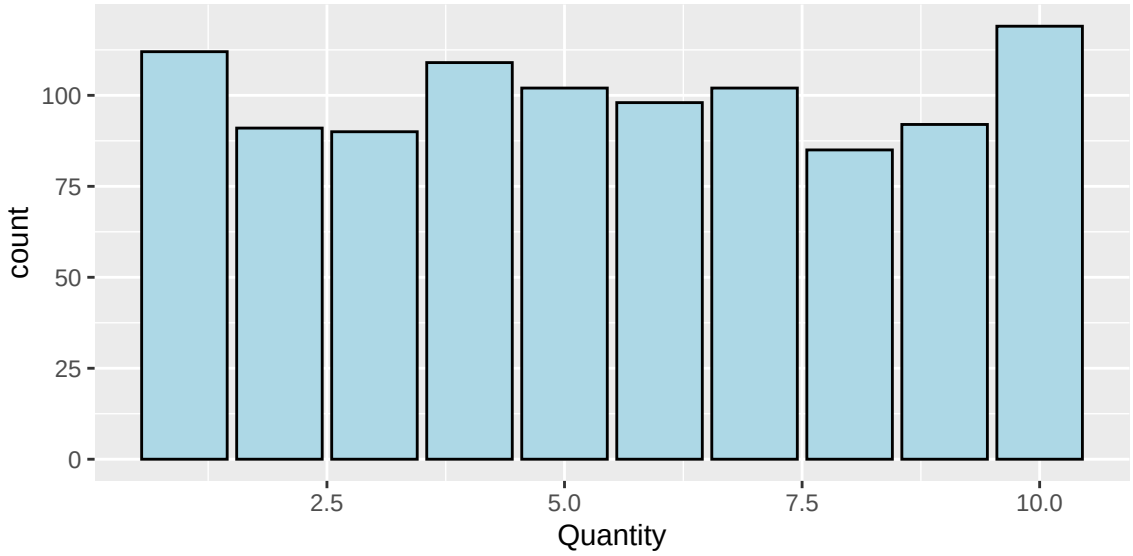
We can fill colors inside the bar graphs. example:

```
ggplot(sdata, aes(x=Quantity)) + geom_bar(fill="lightblue")
```



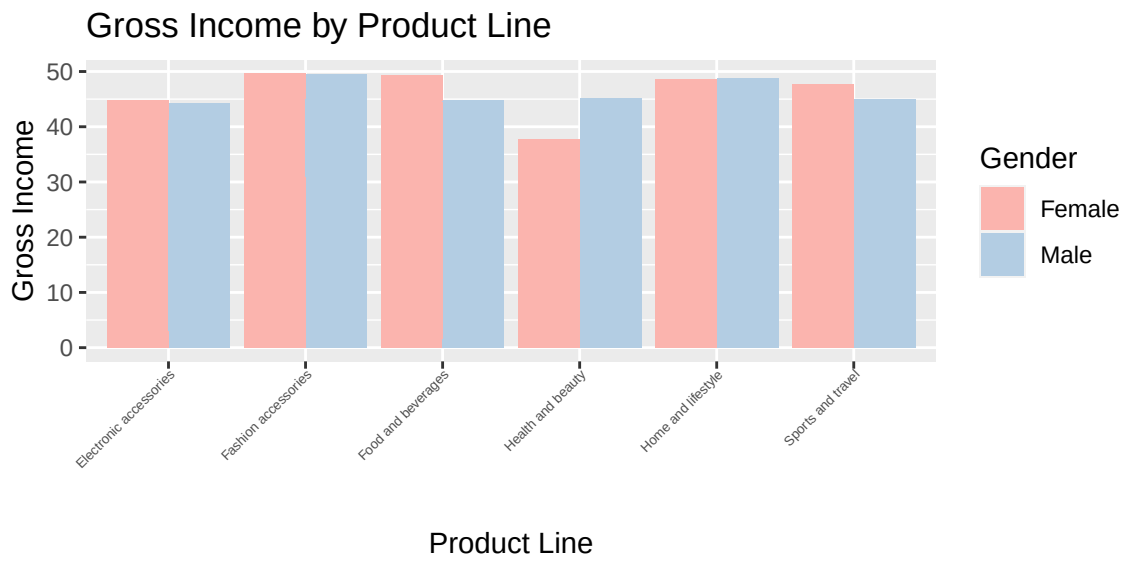
We can also color the border using color().

```
ggplot(sdata, aes(x=Quantity)) + geom_bar(fill="lightblue", color="black")
```



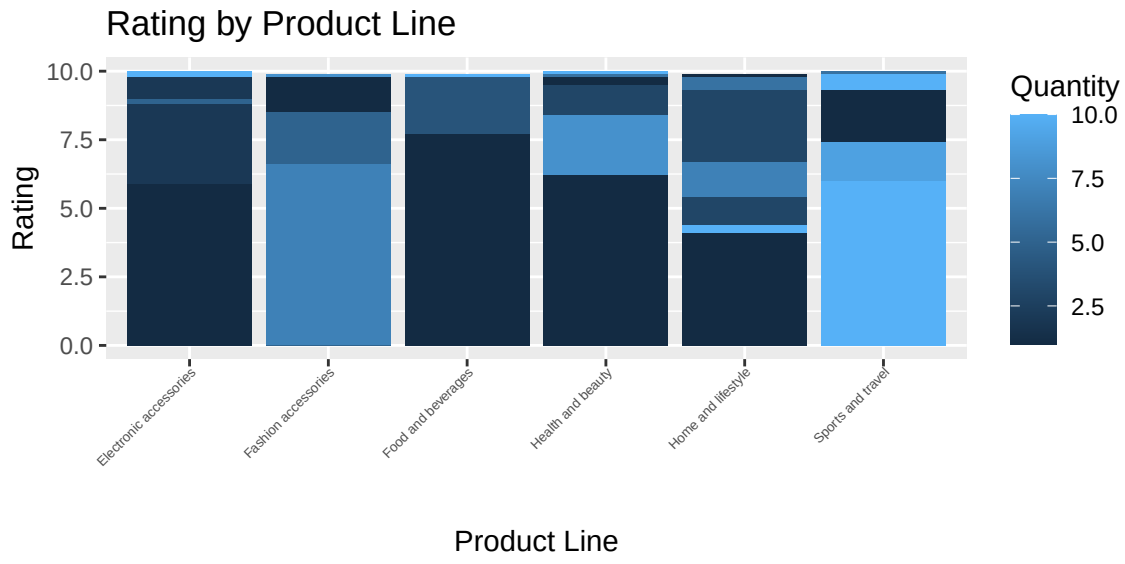
To group bars together, we use position='dodge' in geom_col(). We can apply color changes like palettes.

```
ggplot(sdata, aes(x = Product.line, y = gross.income, fill=Gender)) +
geom_col(position = "dodge") +
labs(x = "Product Line", y = "Gross Income", title = "Gross Income by Product Line") +
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0))) +
scale_fill_brewer(palette = "Pastel1")
```



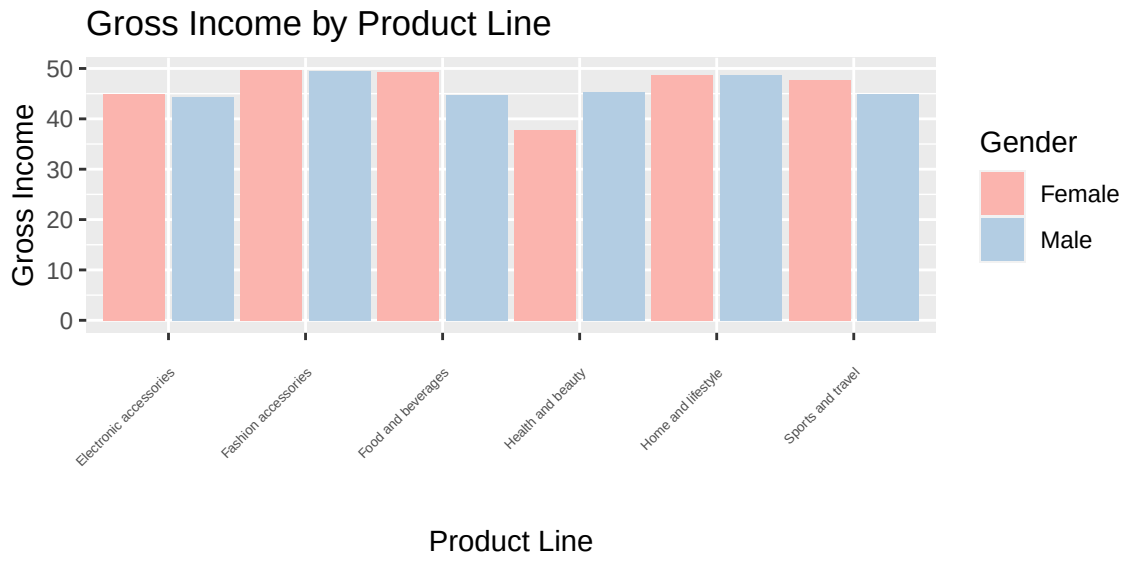
The parameter 'fill' can be used to map with another column in a bar graph.

```
ggplot(sdata, aes(x = Product.line, y = Rating, fill=Quantity)) +
geom_col(position = "dodge") +
labs(x = "Product Line", y = "Rating", title = "Rating by Product Line") +
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



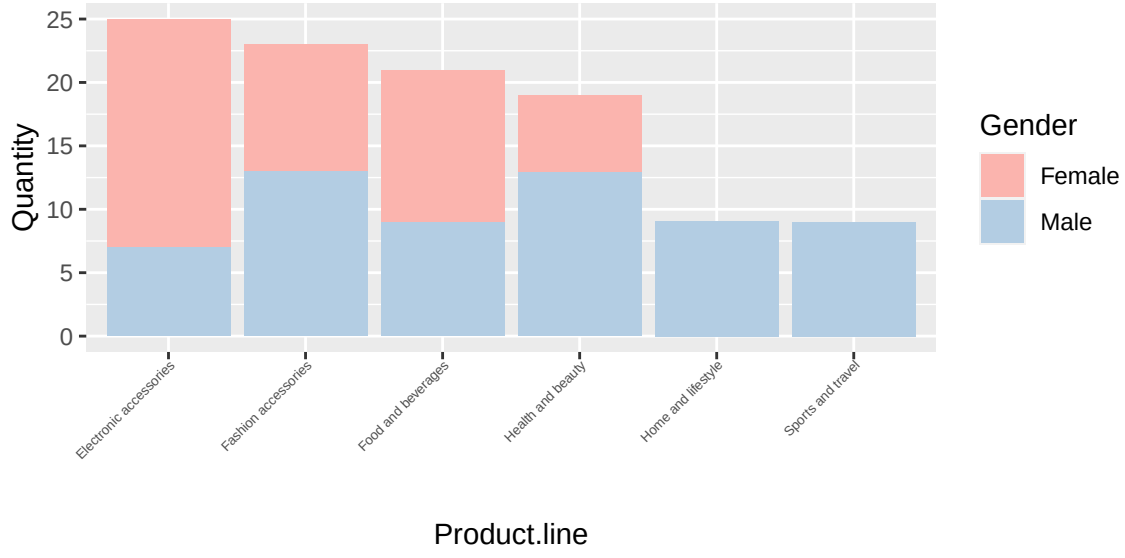
The parameter position = position_dodge() can be used to give space between joint bars.

```
ggplot(sdata, aes(x = Product.line, y = gross.income, fill=Gender)) +
    geom_col(position = position_dodge(1)) +
    labs(x = "Product Line", y = "Gross Income", title = "Gross Income by Product Line") +
    theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
                                     margin = margin(t = 10, r = 10, b = 20, l = 0))) +
    scale_fill_brewer(palette = "Pastel1")
```



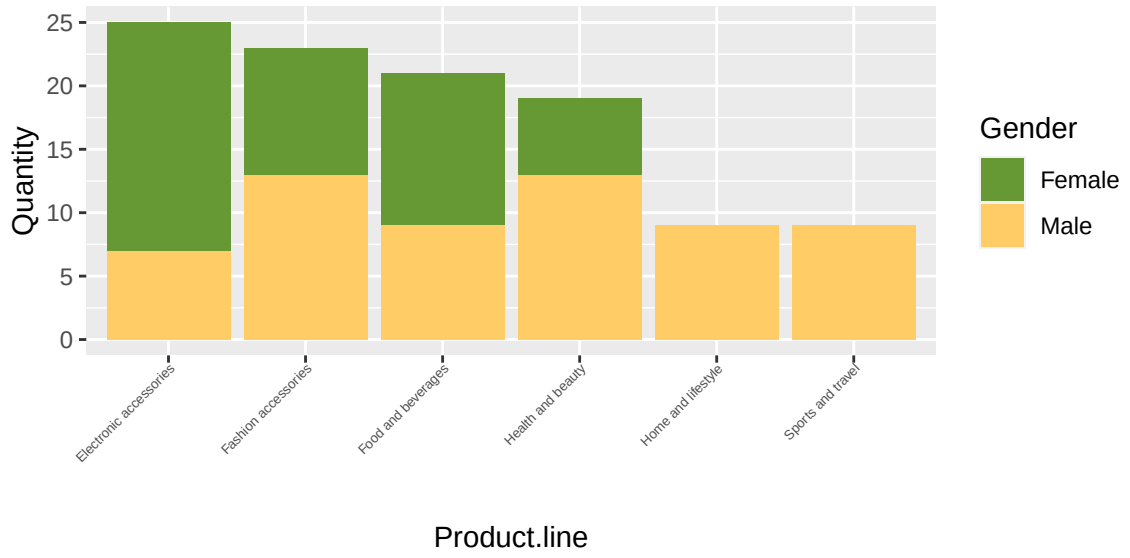
We can use color in bar graph to see a better visualization. Here, we plotted Quantity with Products mapped with Gender.

```
test <- sdata %>% arrange(desc(Unit.price)) %>% slice(1:20)

ggplot(test, aes(y=Quantity, x=Product.line, fill = Gender)) + geom_col() +
   scale_fill_brewer(palette = "Pastel1") + theme(axis.text.x = element_text(angle=45,       hjust=1, s
   margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



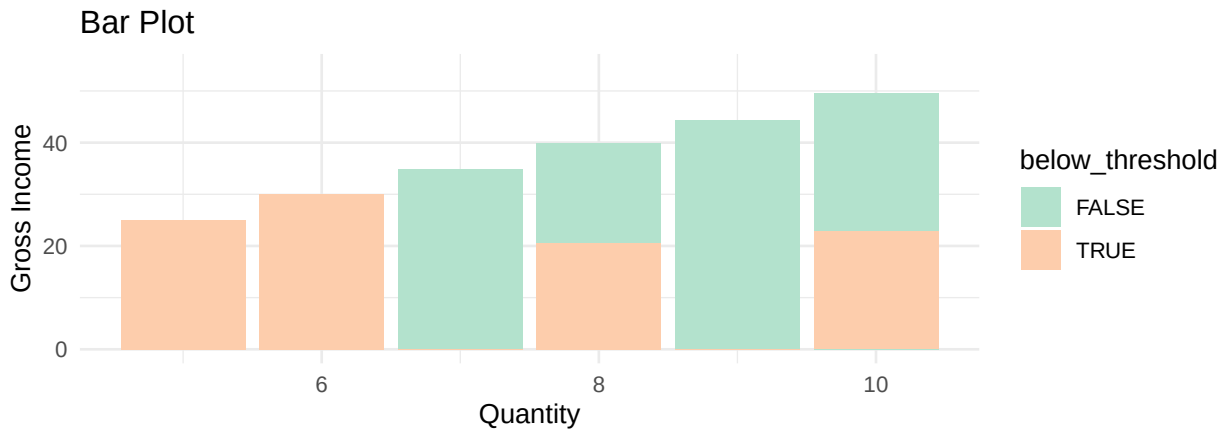
If default colors are not preferred, then colors can be added manually using scale_fill_manual().

```
test <- sdata %>% arrange(desc(Unit.price)) %>% slice(1:20)

ggplot(test, aes(y=Quantity, x=Product.line, fill = Gender)) + geom_col() +                  scale_fi
     margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



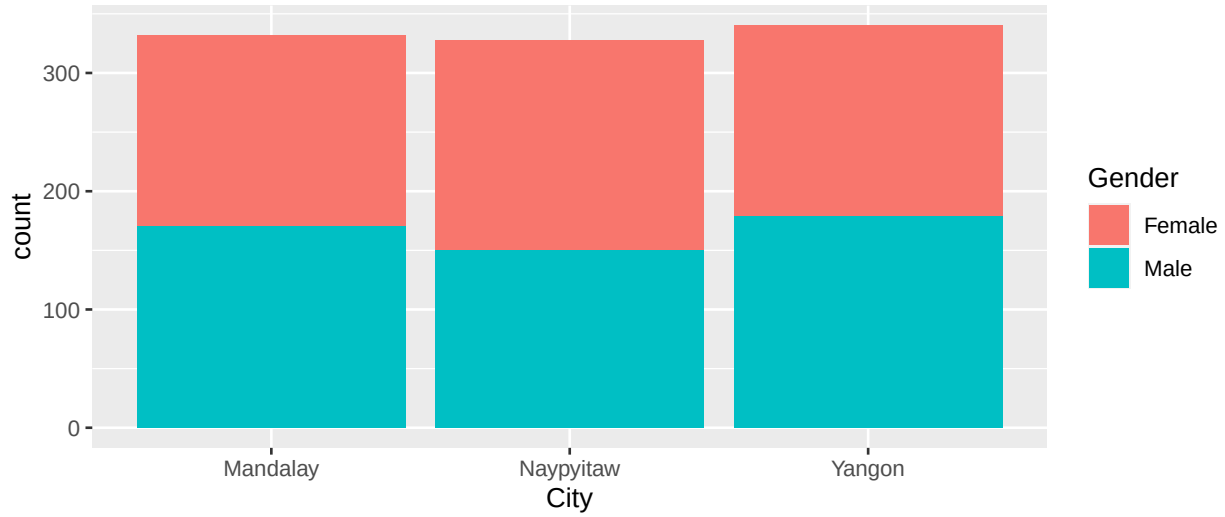
We can use mutate() to create a new column based on our need and use it to fill in the bar graph.

```
data_sub <- sdata %>%
    filter(City == "Yangon" & Quantity >= 5) %>%
    mutate(below_threshold = gross.income < 30)

ggplot(data_sub, aes(x = Quantity, y = gross.income, fill = below_threshold)) +
    geom_col(position = "identity") +
    scale_fill_brewer(palette = "Pastel2") +
    coord_cartesian(ylim = c(0, max(data_sub$gross.income) * 1.1)) +
    labs(x = "Quantity", y = "Gross Income", title = "Bar Plot") +
    theme_minimal()
```



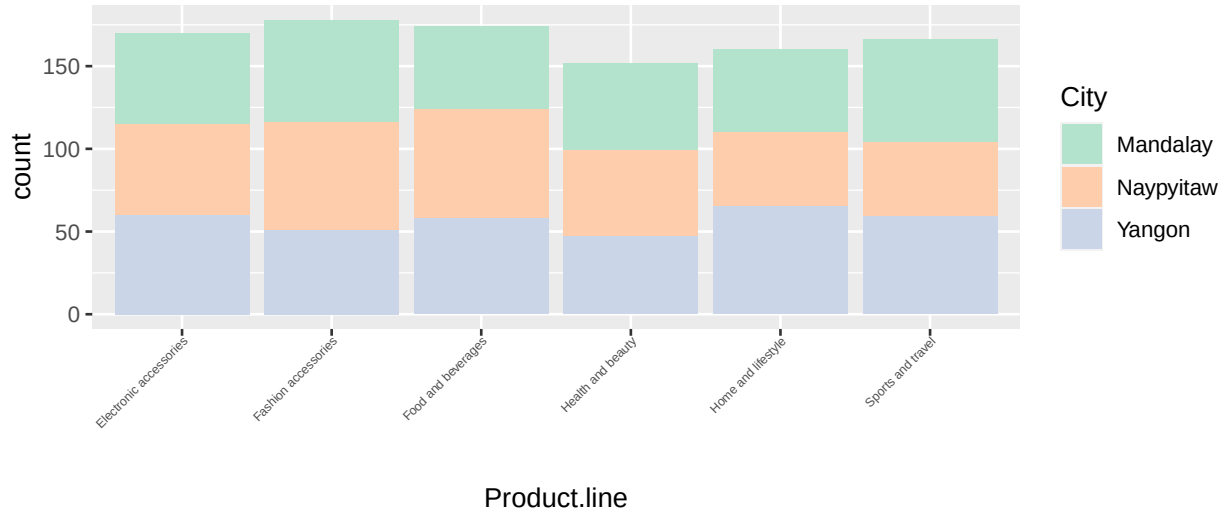
To plot stacked bars, we can use geom_bar() and "fill" parameter. Below graph shows us how many males and females are there in each city.

```
ggplot(sdata, aes(x=City, fill = Gender)) + geom_bar()
```



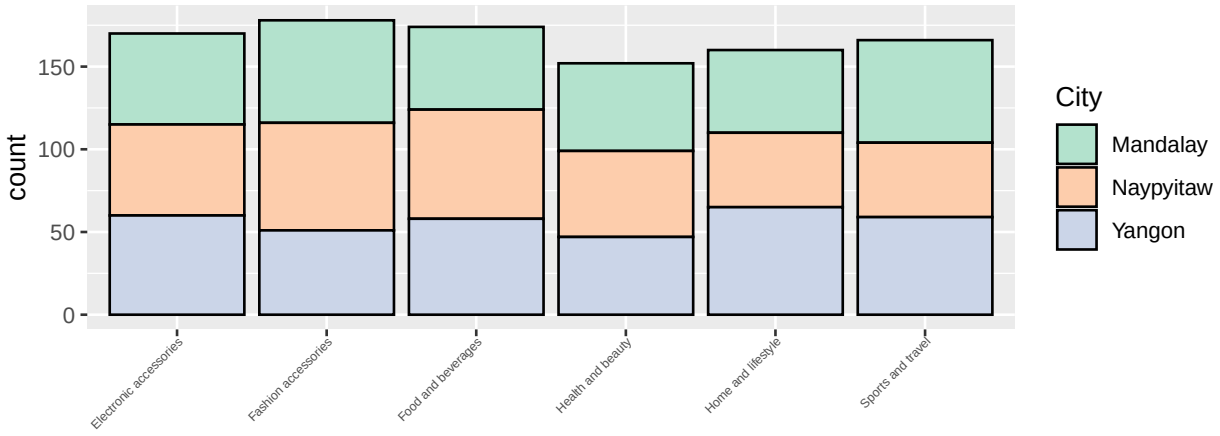
You can even stack multiple bars.

```
ggplot(sdata, aes(x=Product.line, fill = City)) +
geom_bar() +
scale_fill_brewer(palette = "Pastel2") +
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



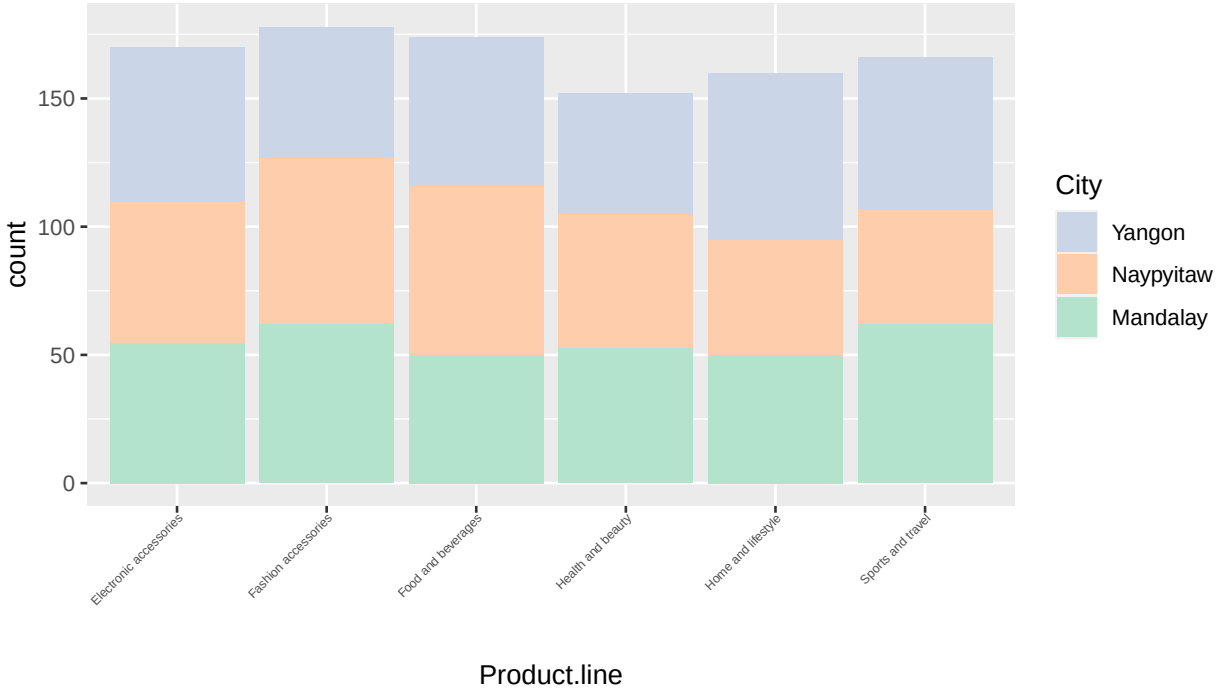
We can outline the bar based on the category which we used to fill for better visualization.

```
ggplot(sdata, aes(x=Product.line, fill = City)) +
geom_bar(color = 'black') +
scale_fill_brewer(palette = "Pastel2") +
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



You can reverse the legend by switching the stack.

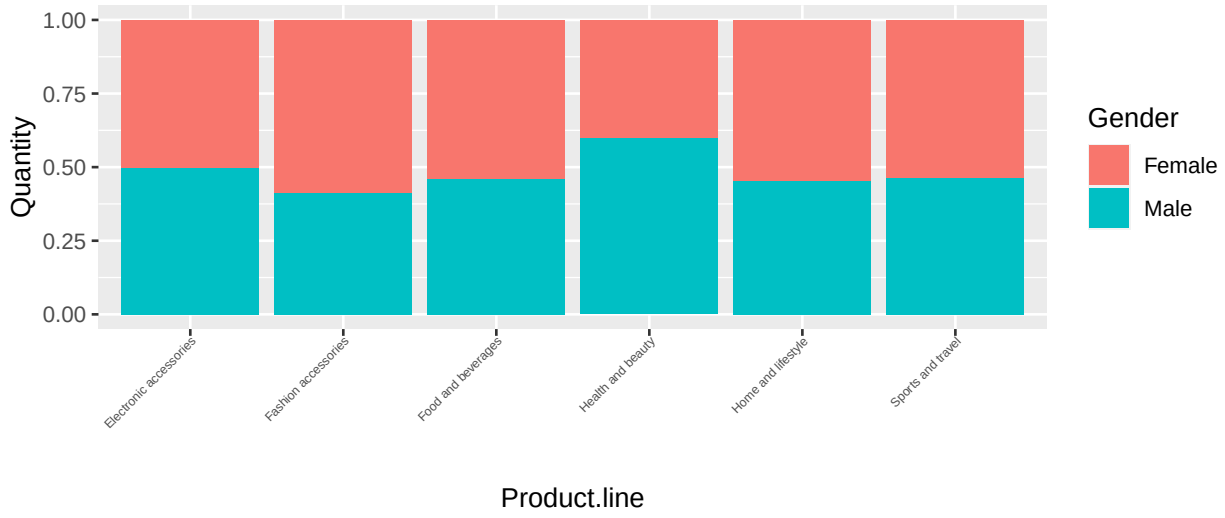
```
ggplot(sdata, aes(x=Product.line, fill = City)) +
geom_bar(position = position_stack(reverse=TRUE)) +
guides(fill = guide_legend(reverse = TRUE)) +
scale_fill_brewer(palette = "Pastel2") +
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



To make a Proportional Stack Bar Graph, we use position="fill" parameter.

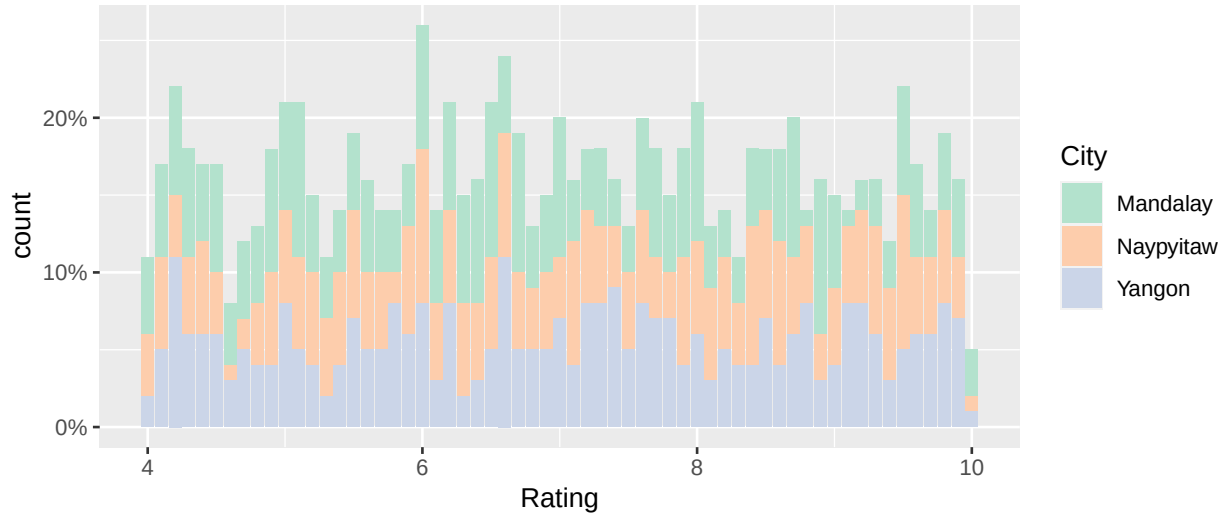
```
ggplot(sdata, aes(x=Product.line, y=Quantity, fill = Gender)) +
geom_col(position="fill") +
```

```
theme(axis.text.x = element_text(angle=45, hjust=1, size = 5,
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



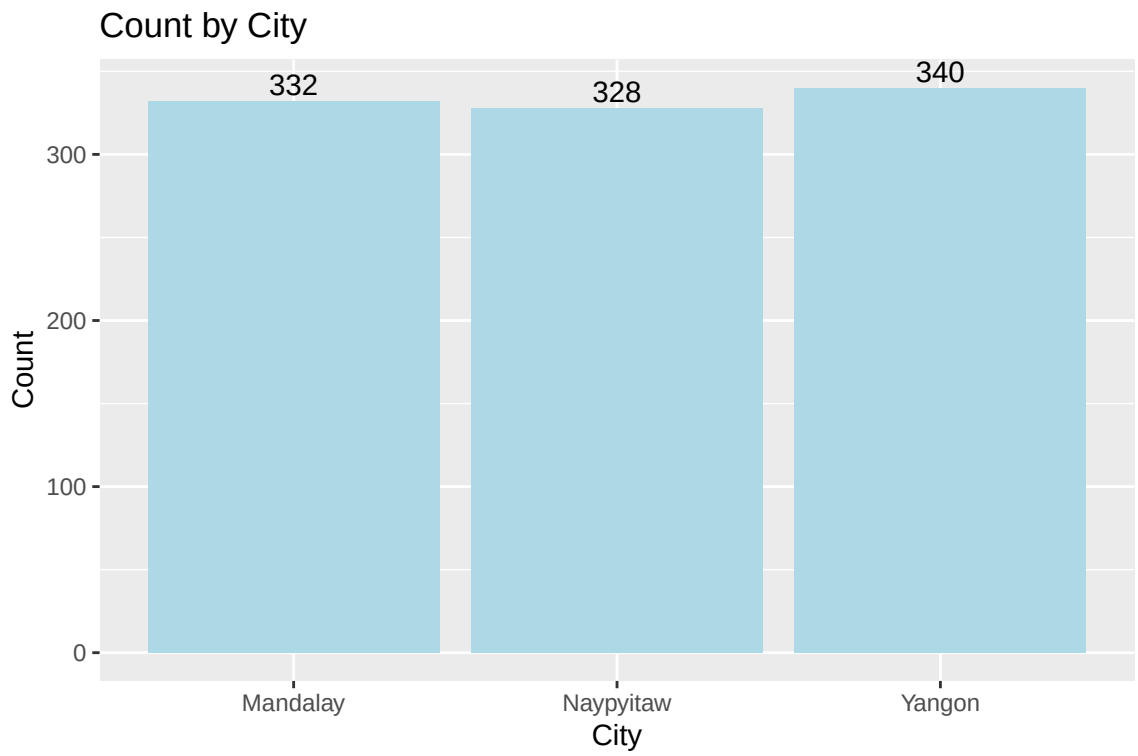
We can also view the axes in percentage.

```
ggplot(sdata, aes(x=Rating, fill = City)) +
    geom_bar() +
    scale_fill_brewer(palette = "Pastel2") +
    scale_y_continuous(labels = scales::percent_format(scale=1))
```



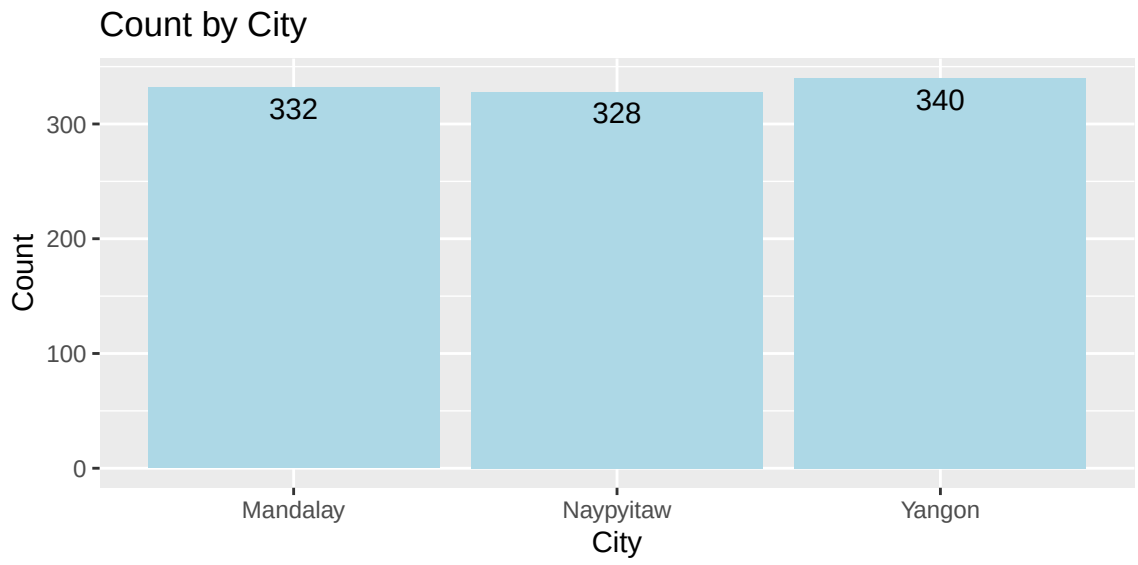
We can assign labels to the bars using geom_text(). Here label will be below the top.

```
ggplot(sdata, aes(x = City)) +
geom_bar(fill="lightblue") +
geom_text(stat = "count", aes(label = after_stat(count)), vjust = -0.3, color = "black") +
labs(x = "City", y = "Count", title = "Count by City")
```



Here label will be above the top.

```
ggplot(sdata, aes(x = City)) +
geom_bar(fill="lightblue") +
geom_text(stat = "count", aes(label = after_stat(count)), vjust = 1.5, color = "black") +
labs(x = "City", y = "Count", title = "Count by City")
```

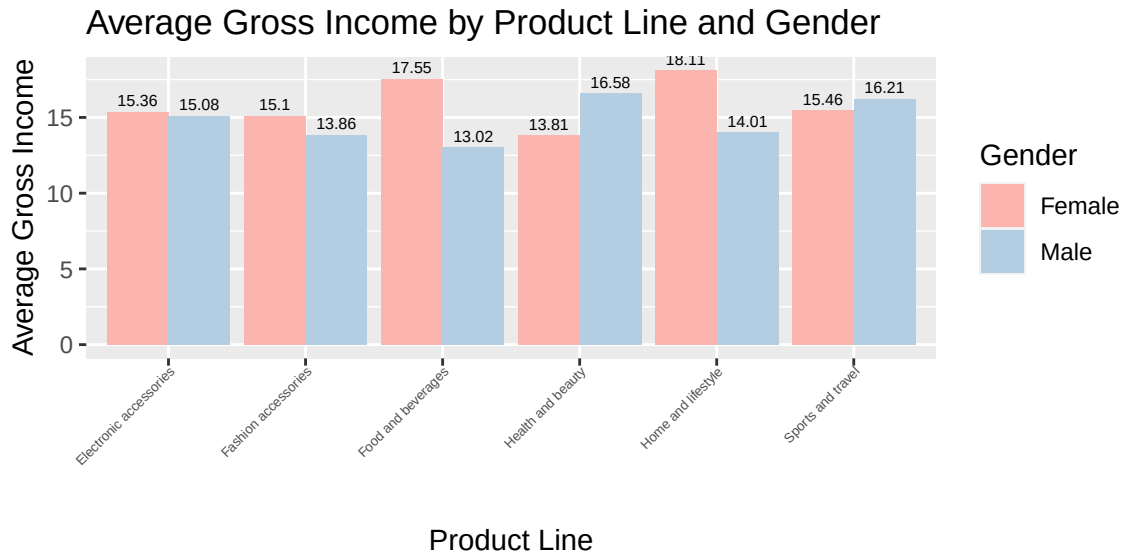


We can label joint bars too. Below is an example of avg gross income of each gender based on product line.

```
avg_income <- sdata %>%
    group_by(Product.line, Gender) %>%
    summarise(avg_income = mean(gross.income))
```

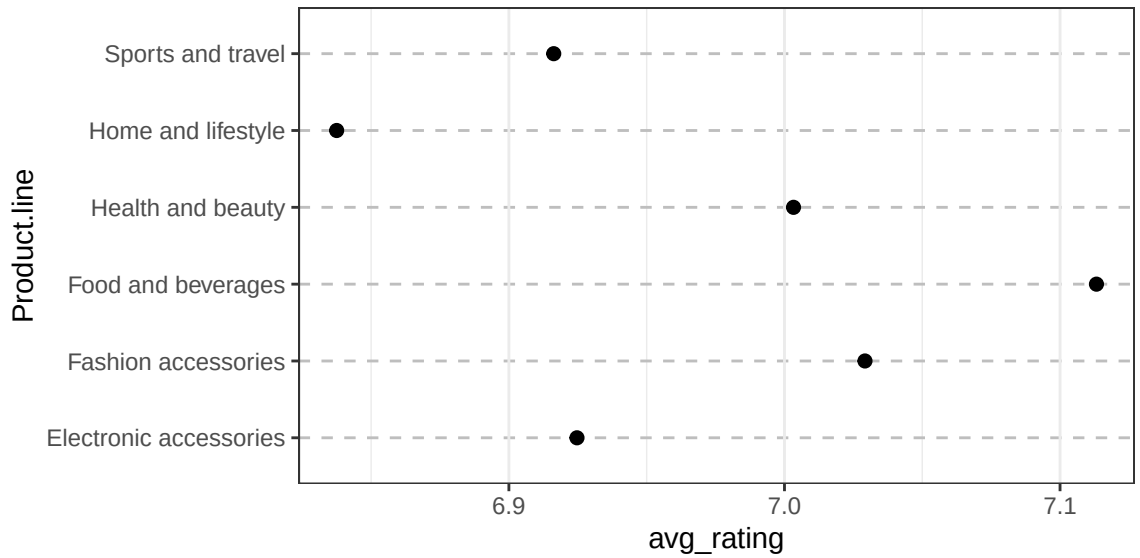
```
## `summarise()` has grouped output by 'Product.line'. You can override using the
## `.groups` argument.
```

```
ggplot(avg_income, aes(x = Product.line, y = avg_income, fill = Gender)) + geom_bar(stat = "identity", p
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```



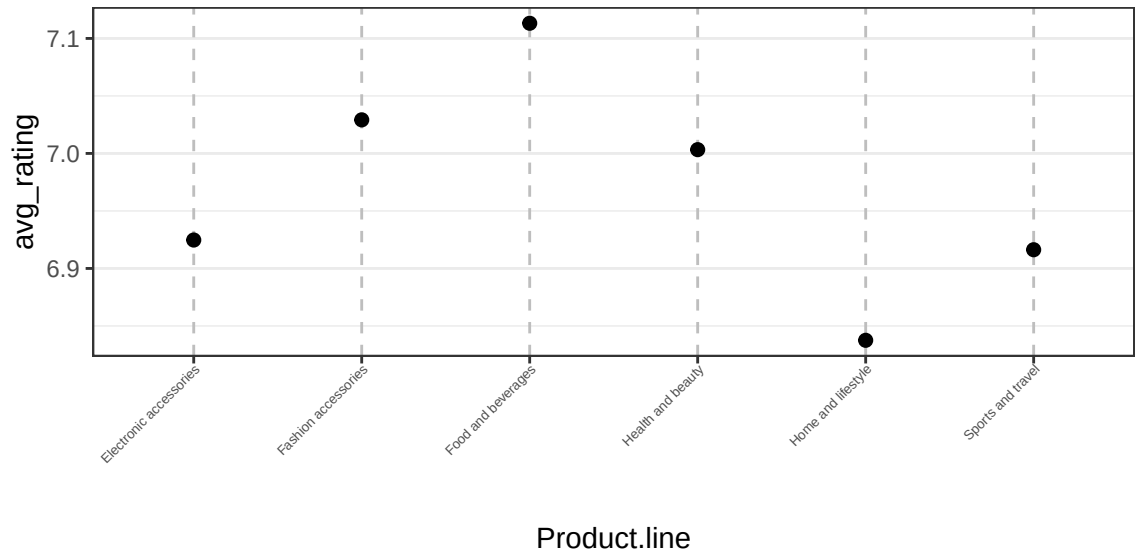
The below graph is a Cleveland Dot Plot which can be created using geom_point().

```
data2 <- sdata %>% group_by(Product.line) %>% summarise(avg_rating = mean(Rating))

ggplot(data2, aes(x=avg_rating, y = Product.line)) + geom_point(size=2) + theme_bw() + theme(panel.grid
```

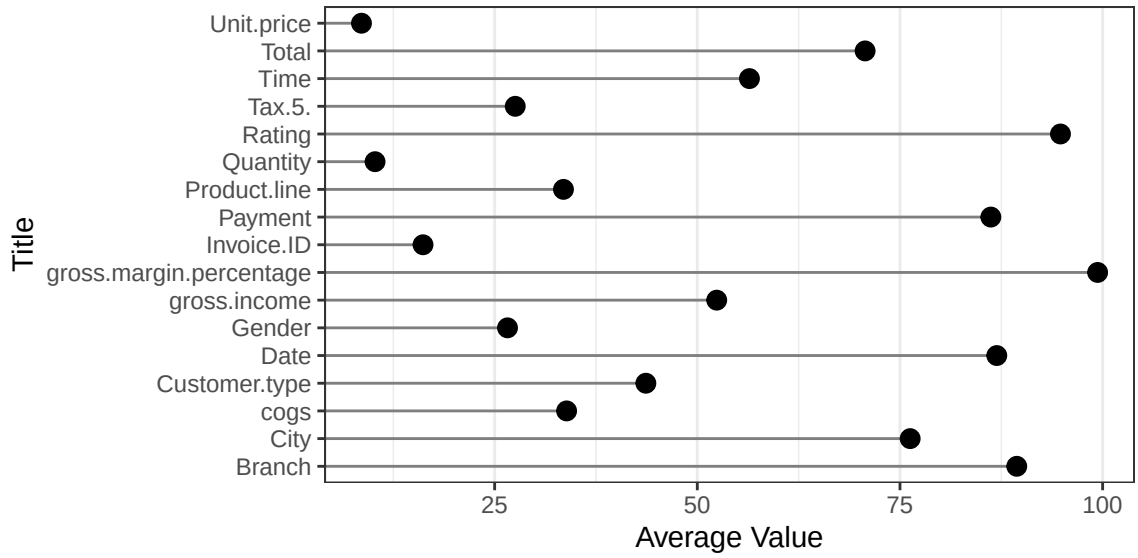


We can also change the axes for the above graph to see a different perspective.

```
ggplot(data2, aes(y=avg_rating, x = Product.line)) + geom_point(size=2) + theme_bw() + theme(panel.grid
margin = margin(t = 0, r = 0, b = 20, l = 0)))
```

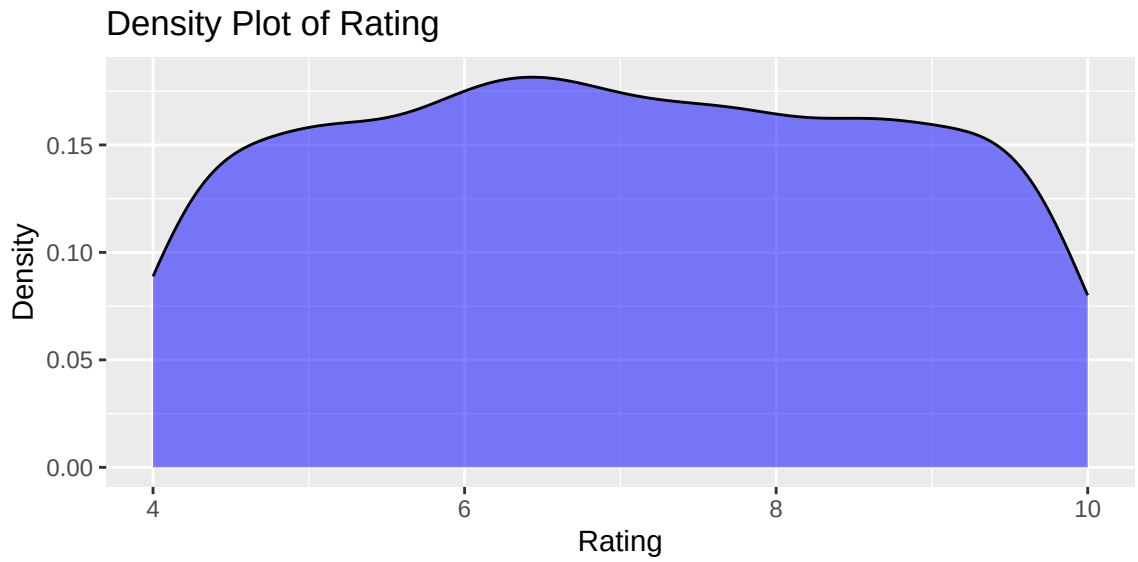



```
titles <- c("Invoice.ID", "Branch", "City", "Customer.type", "Gender",
            "Product.line", "Unit.price", "Quantity", "Tax.5.",
            "Total", "Date", "Time", "Payment", "cogs",
            "gross.margin.percentage", "gross.income", "Rating")

# Create a sample data frame with random values
data <- data.frame(
    title = titles,
    avg = runif(length(titles), min = 0, max = 100)
)

data <- data[order(data$avg), ]

# Plot the Cleveland dot plot
ggplot(data, aes(x = avg, y = title)) +
    geom_segment(aes(yend = title), xend = 0, colour = "grey50") +
    geom_point(size = 3) +
    theme_bw() +
    labs(x = "Average Value", y = "Title") +
    theme(
        panel.grid.major.y = element_blank(),  # No horizontal grid lines
        legend.position = "none"
    )
```



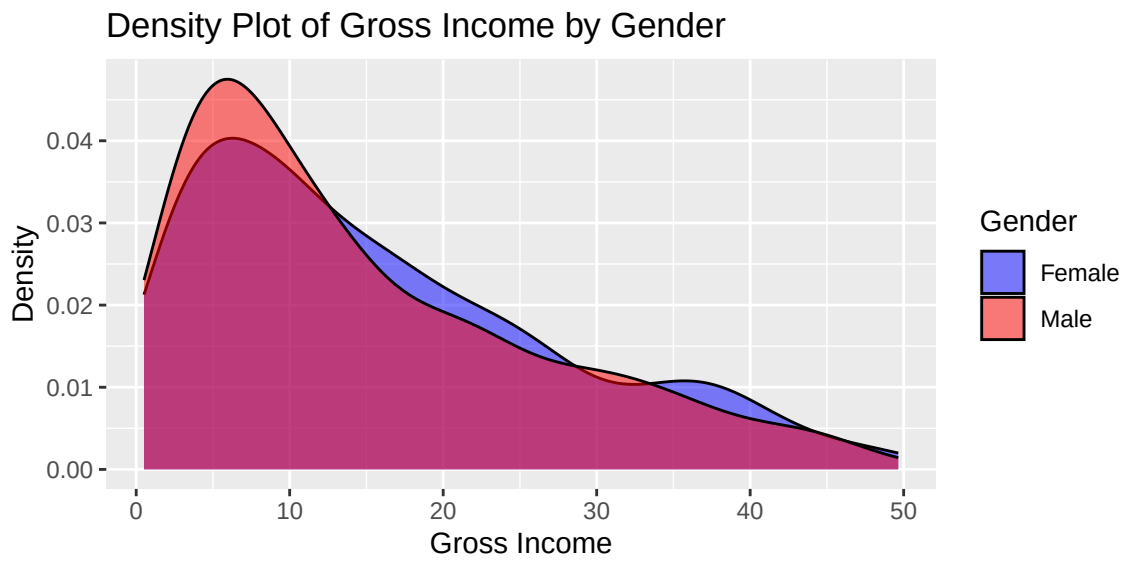
We can plot a density graph using geom_density(). Below are the examples.

```
ggplot(sdata, aes(x = Rating)) +
    geom_density(fill = "blue", alpha = 0.5) +
    labs(x = "Rating", y = "Density", title = "Density Plot of Rating")
```



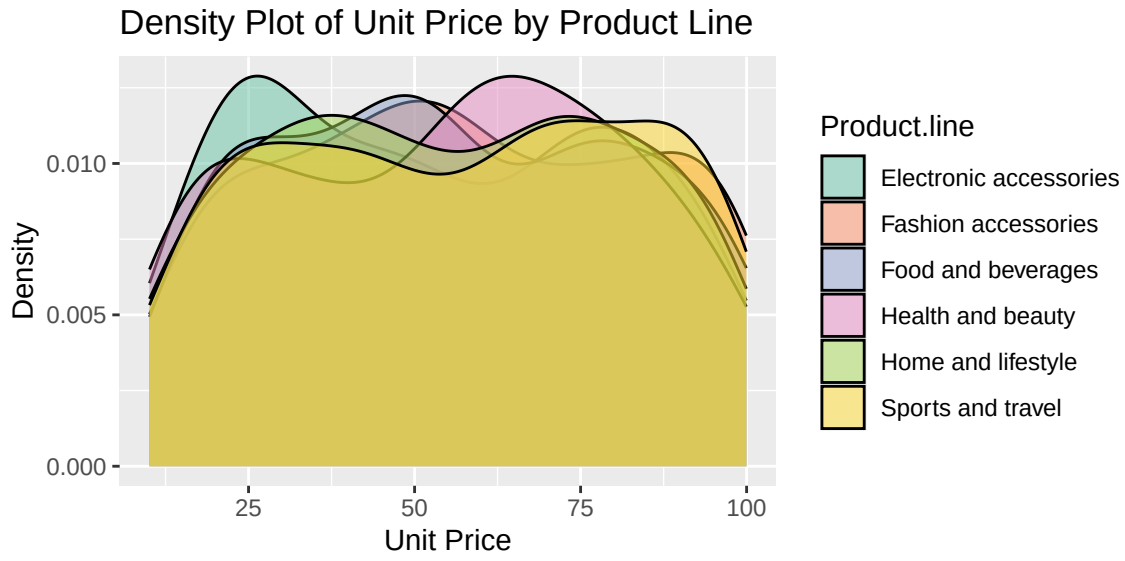
This graph shows Density plot of Gross income by Gender.

```
ggplot(sdata, aes(x = gross.income, fill = Gender)) +
    geom_density(alpha = 0.5) +
    labs(x = "Gross Income", y = "Density", title = "Density Plot of Gross Income by Gender") +
    scale_fill_manual(values = c("Female" = "blue", "Male" = "red"))
```



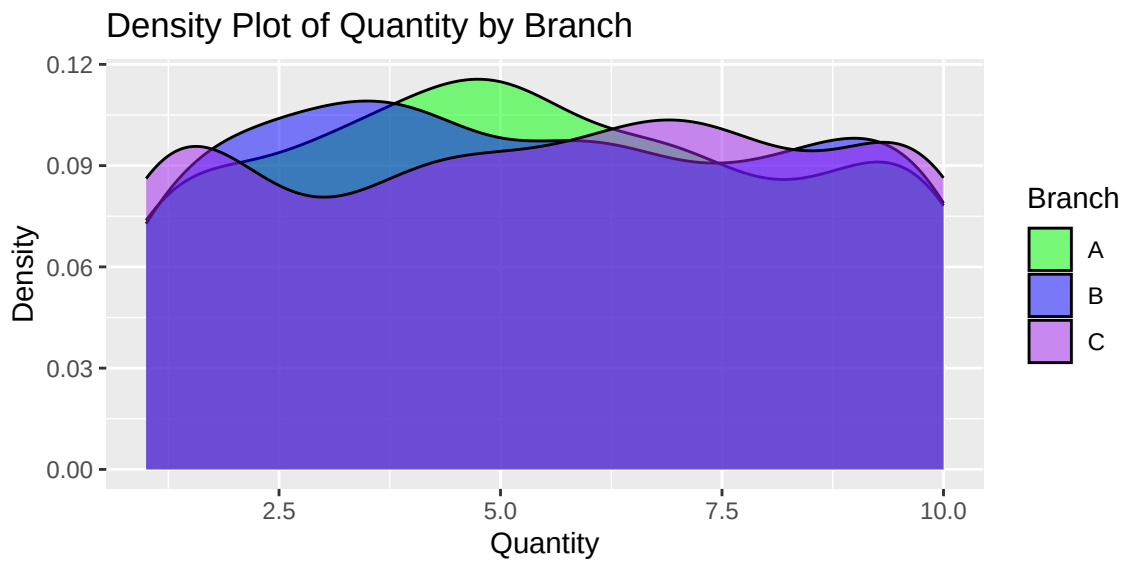
This graph describes Density plot of price by product.

```
ggplot(sdata, aes(x = Unit.price, fill = Product.line)) +
  geom_density(alpha = 0.5) +
  labs(x = "Unit Price", y = "Density", title = "Density Plot of Unit Price by Product Line") +
  scale_fill_brewer(palette = "Set2")
```



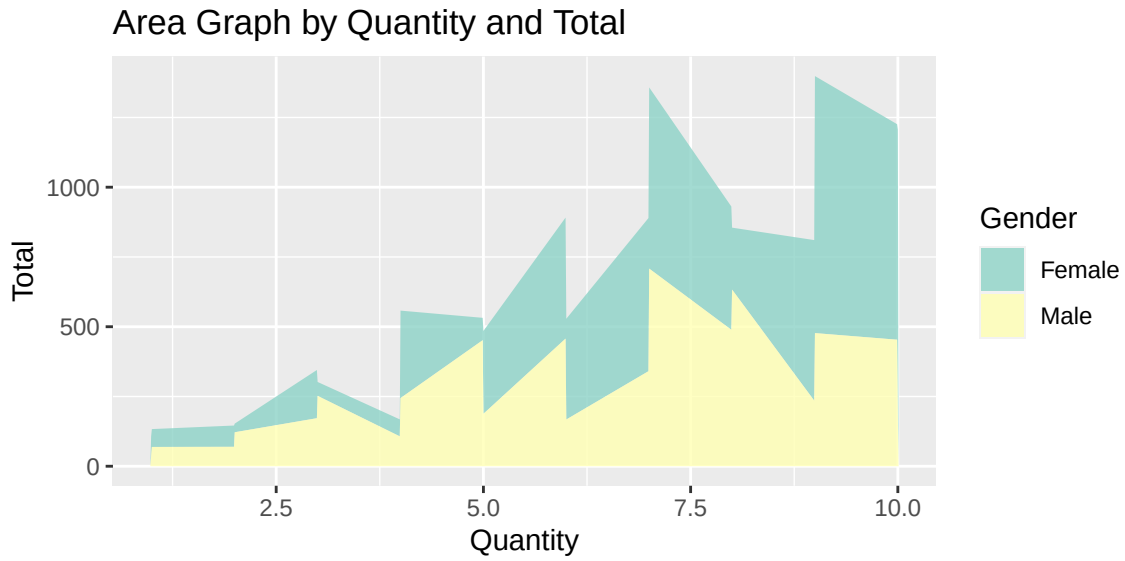
The graph below describes density of Quantity per Branch.

```
ggplot(sdata, aes(x = Quantity, fill = Branch)) +
    geom_density(alpha = 0.5) +
    labs(x = "Quantity", y = "Density", title = "Density Plot of Quantity by Branch") +
    scale_fill_manual(values = c("A" = "green", "B" = "blue", "C" = "purple"))
```



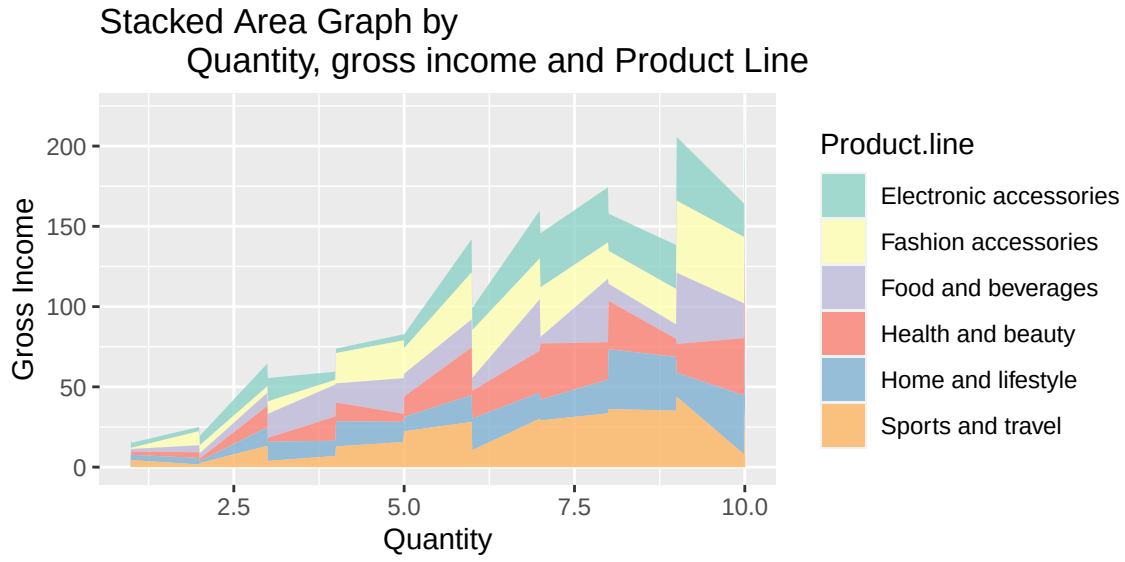
We can plot area graphs using geom_area().

```
ggplot(sdata, aes(fill = Gender, x = Quantity, y = Total)) +
    geom_area(alpha = 0.8) +
    labs(y = "Total", x = "Quantity", title = "Area Graph by Quantity and Total") +
    scale_fill_brewer(palette = "Set3")
```



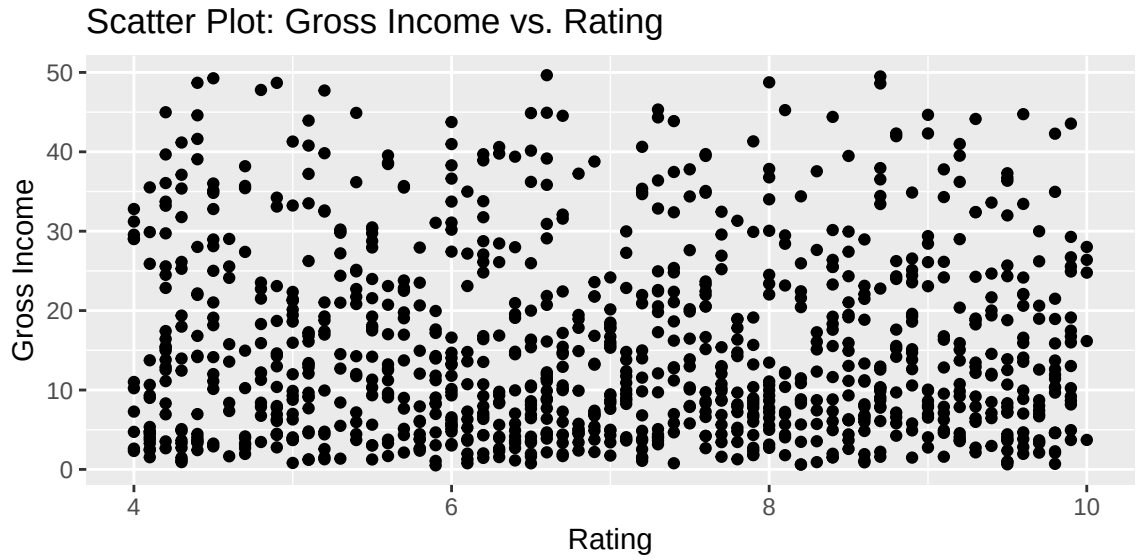
We can create stacked area graphs using geom_area() and fill it with whatever Category as required.

```
ggplot(sdata, aes(y = gross.income, x = Quantity, fill = Product.line)) +
    geom_area(position = "stack", alpha = 0.8) +
    labs(y = "Gross Income", x = "Quantity", title = "Stacked Area Graph by
         Quantity, gross income and Product Line") +
    scale_fill_brewer(palette = "Set3")
```



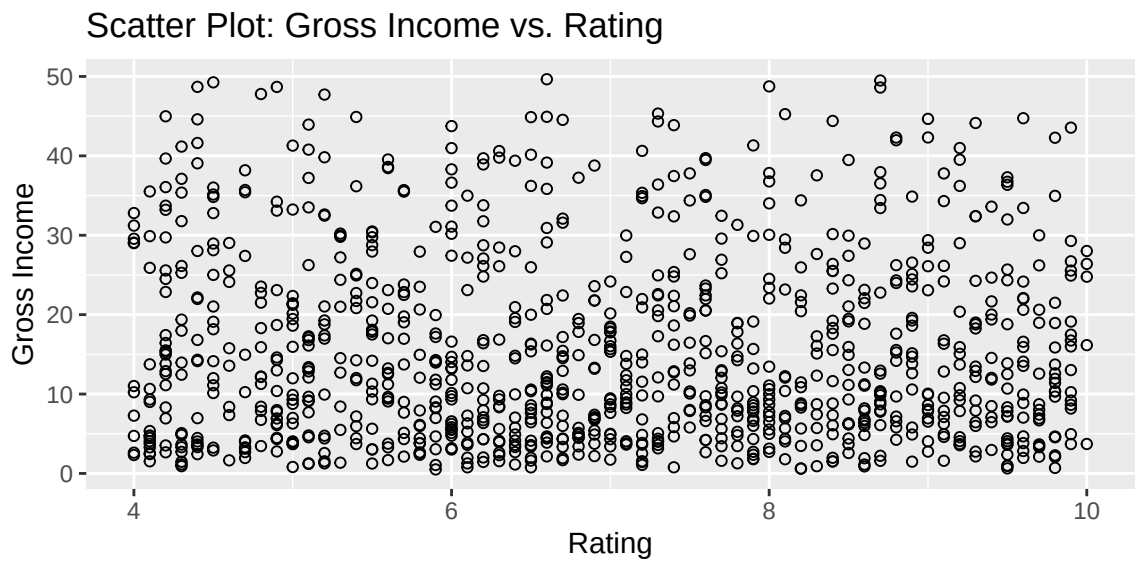
We can produce scatter plots using geom_point().

```
ggplot(sdata, aes(x = Rating, y = gross.income)) +
    geom_point() +
    labs(x = "Rating", y = "Gross Income", title = "Scatter Plot: Gross Income vs. Rating")
```



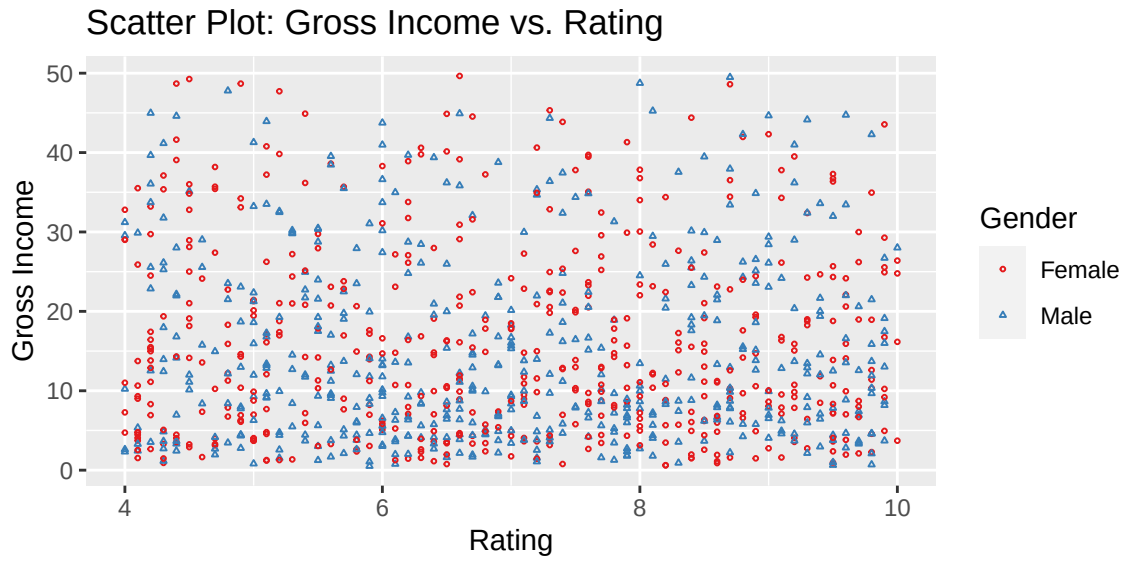
We can change the shapes of the points too.

```
ggplot(sdata, aes(x = Rating, y = gross.income)) +
    geom_point(shape = 21) +
    labs(x = "Rating", y = "Gross Income", title = "Scatter Plot: Gross Income vs. Rating")
```



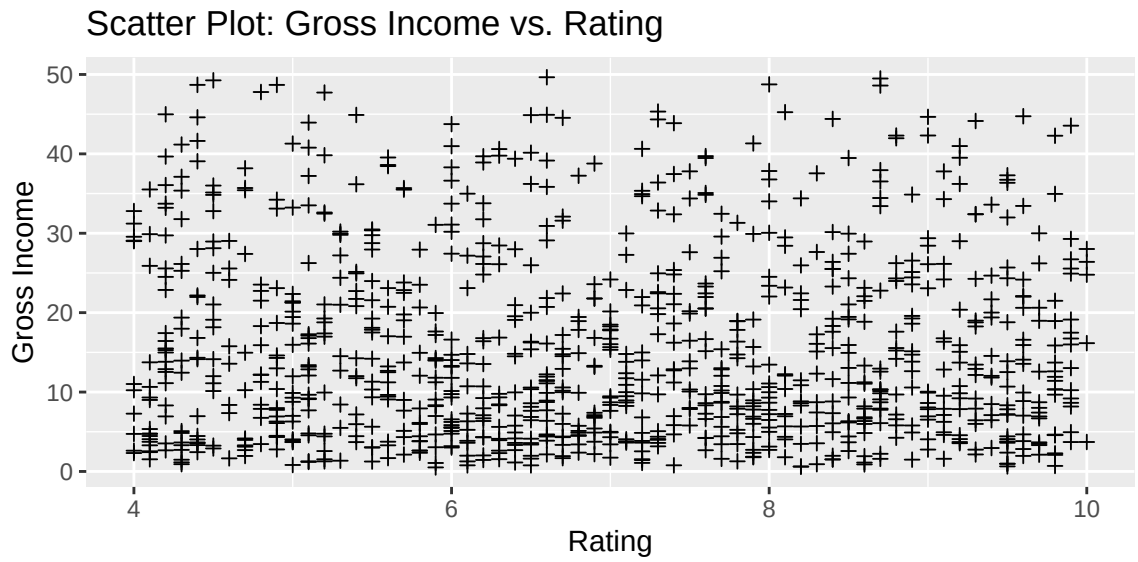
We can make a scatter plot with different colors and shapes too.

```
ggplot(sdata, aes(x = Rating, y = gross.income, color = Gender, shape = Gender)) +
    geom_point(size = 0.5) +
    labs(x = "Rating", y = "Gross Income", title = "Scatter Plot: Gross Income vs. Rating") +
    scale_shape_manual(values = c(1,2)) +
    scale_color_brewer(palette = "Set1")
```



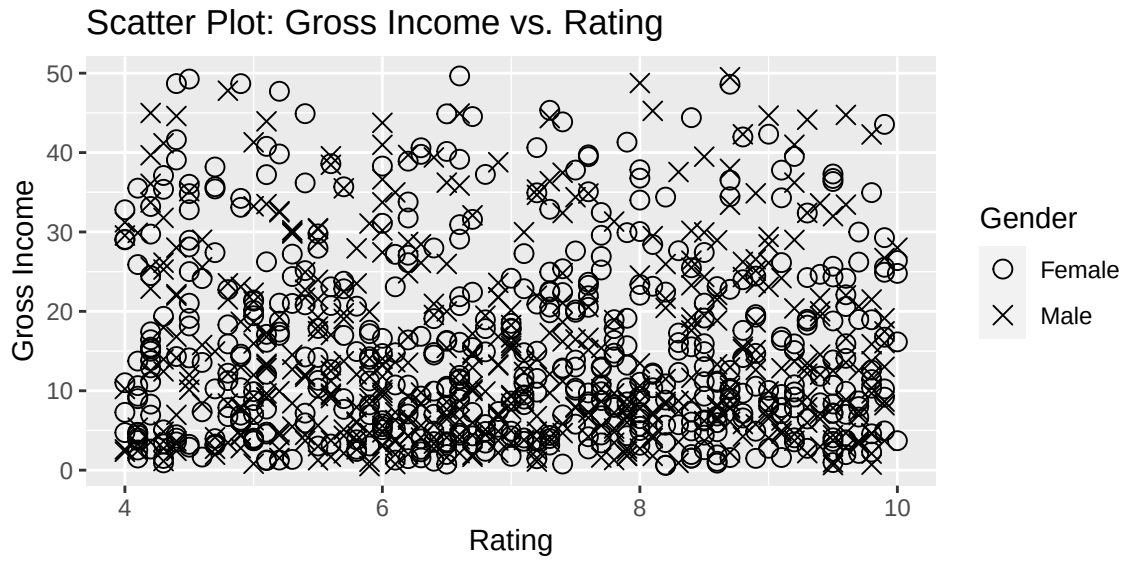
We can change the shapes used to represent the data.

```
ggplot(sdata, aes(x = Rating, y = gross.income)) +
    geom_point(shape = 3) +
    labs(x = "Rating", y = "Gross Income", title = "Scatter Plot: Gross Income vs. Rating") +
    scale_shape_manual(values = c(1,2)) +
    scale_color_brewer(palette = "Set1")
```



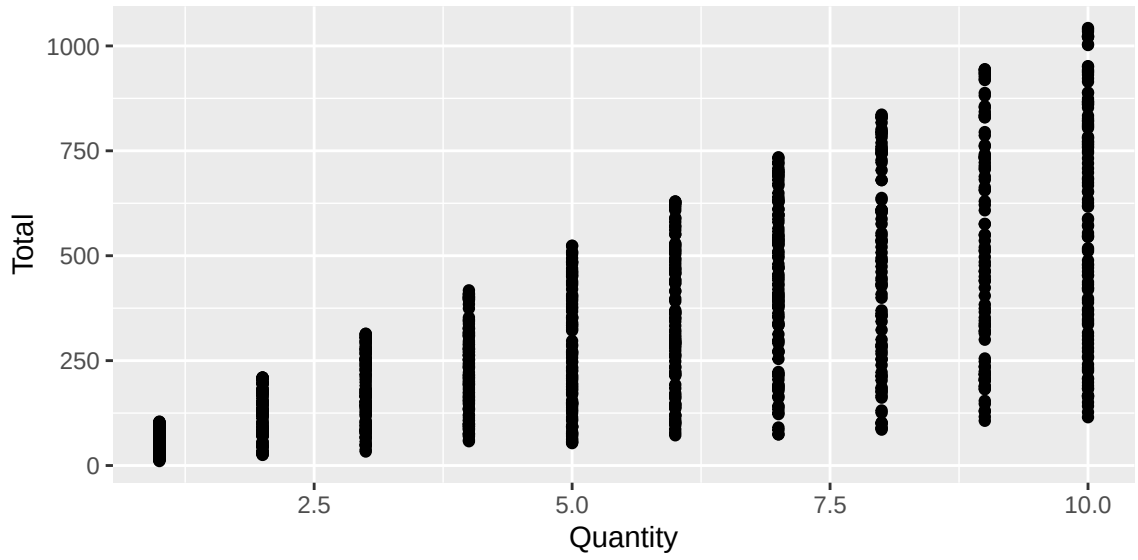
We can represent points with different icons based on another category.

```
ggplot(sdata, aes(x = Rating, y = gross.income, shape = Gender)) +
    geom_point(size = 3) +
    labs(x = "Rating", y = "Gross Income", title = "Scatter Plot: Gross Income vs. Rating") +
    scale_shape_manual(values = c(1,4)) +
    scale_color_brewer(palette = "Set1")
```

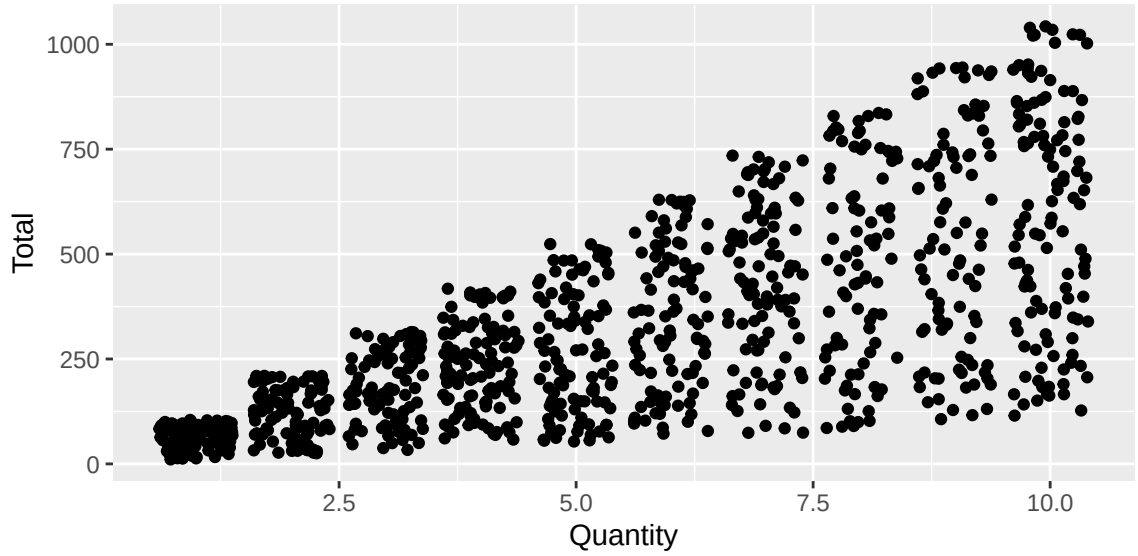


We can represent Discrete variable scatter plots better using jitter.

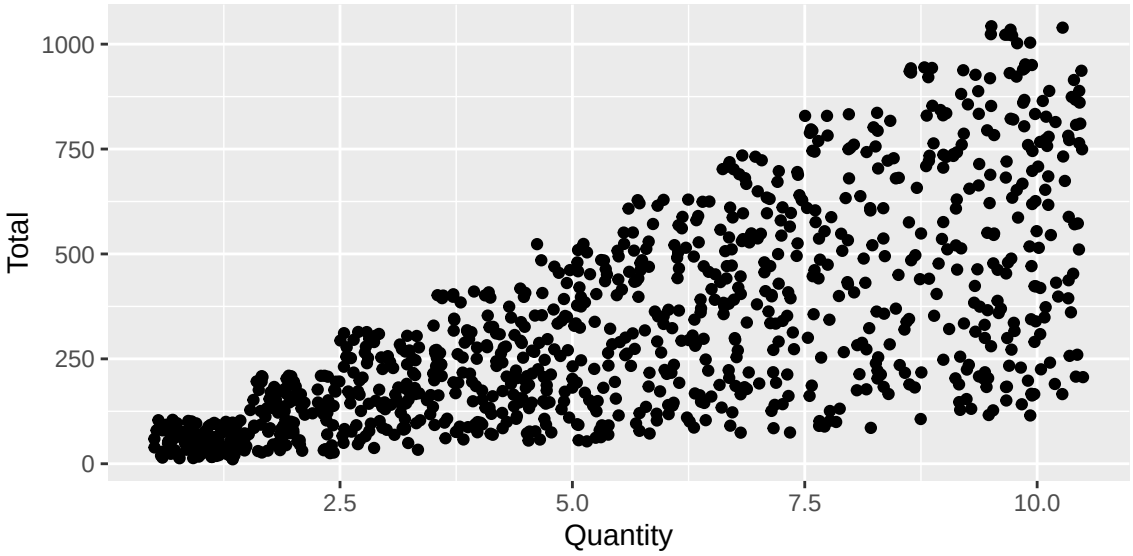
```
ggplot(sdata, aes(x=Quantity, y=Total)) + geom_point()
```




```
ggplot(sdata, aes(x=Quantity, y=Total)) + geom_point(position = "jitter")
```



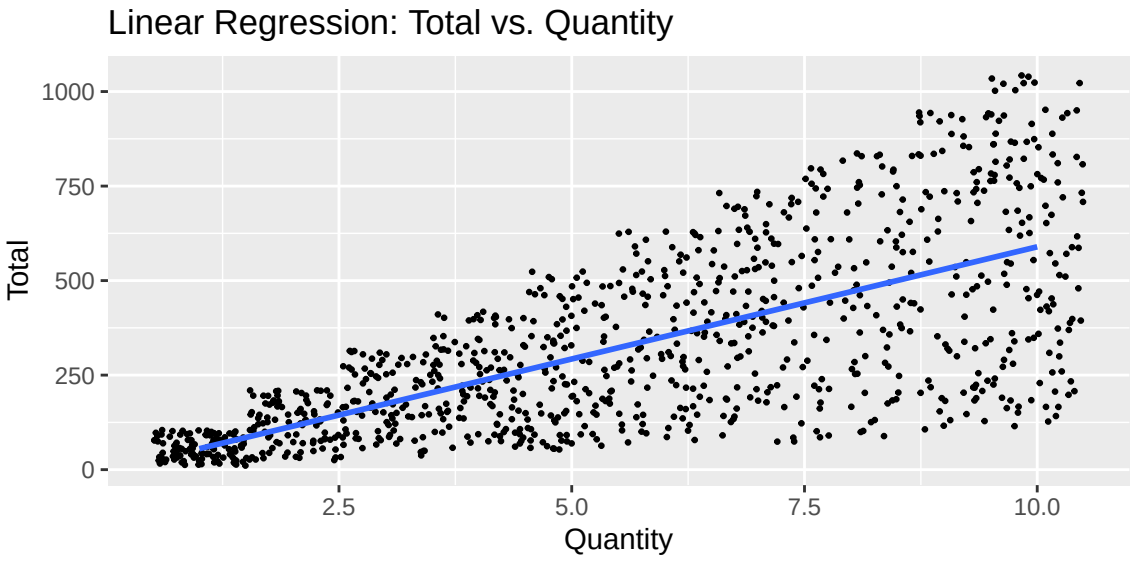

```
ggplot(sdata, aes(x=Quantity, y=Total)) +
  geom_point(position = position_jitter(width = .5, height = 0))
```



We use geom_smooth() and `method = "lm"` to create a linear regression plot.

```
lm_total_quantity <- lm(Total ~ Quantity, data = sdata)

ggplot(sdata, aes(x = Quantity, y = Total)) +
    geom_point(position = position_jitter(width = .5, height = 0), size = 0.5) +
    geom_smooth(method = "lm", se = FALSE) +
    labs(x = "Quantity", y = "Total", title = "Linear Regression: Total vs. Quantity")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

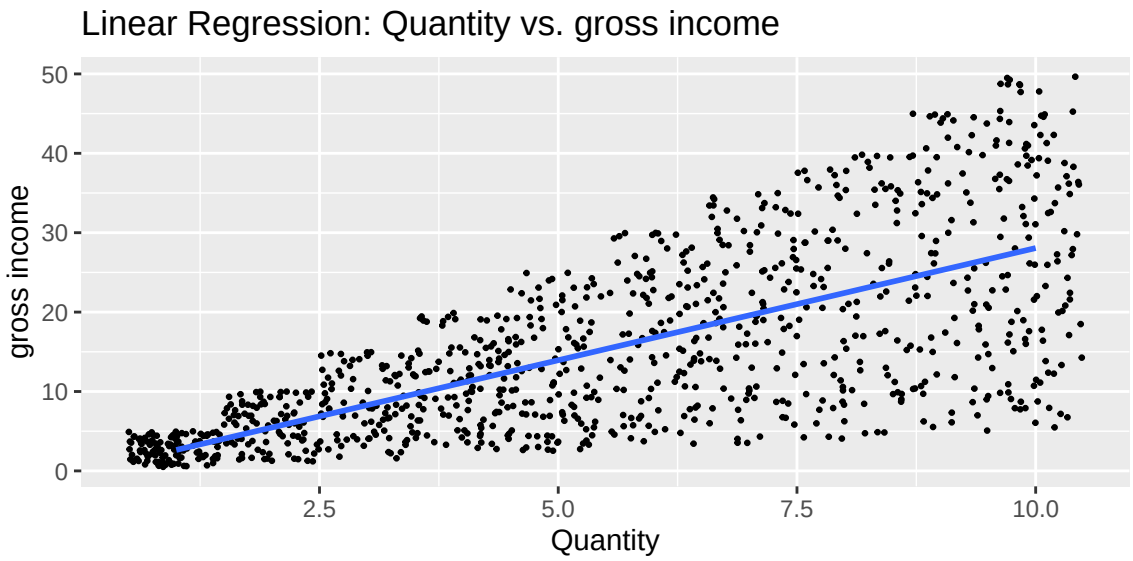


Here are more linear regression plots.

```
lm_total_quantity <- lm(Quantity ~ gross.income, data = sdata)
```

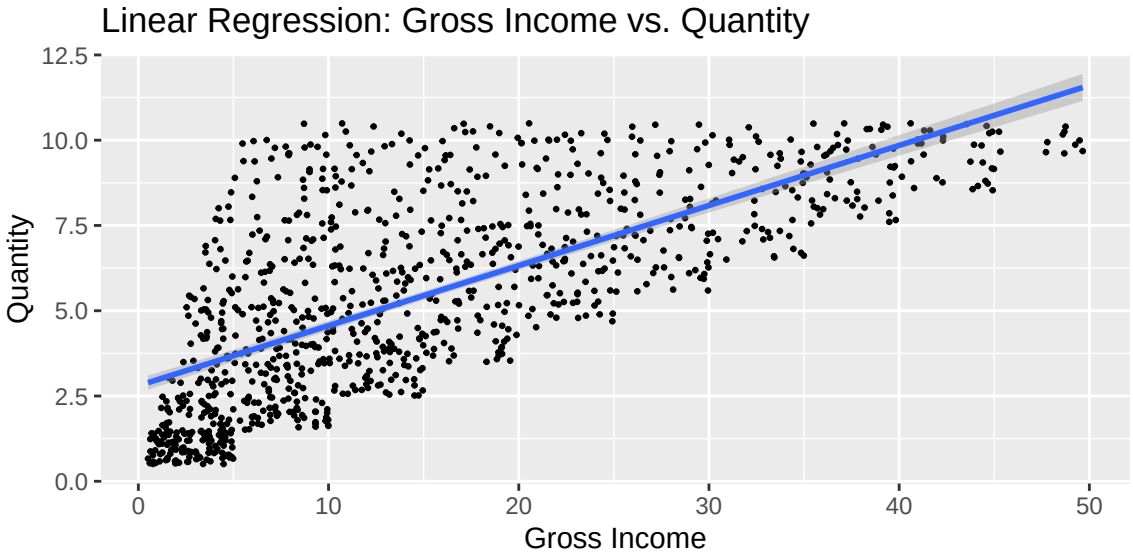
```
ggplot(sdata, aes(x = Quantity, y = gross.income)) +
    geom_point(position = position_jitter(width = .5, height = 0), size = 0.5) +
    geom_smooth(method = "lm", se = FALSE) +
    labs(x = "Quantity", y = "gross income", title = "Linear Regression: Quantity vs. gross income")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(sdata, aes(y = Quantity, x = gross.income)) +
    geom_point(position = position_jitter(width = 0, height = 0.5), size = 0.5) +
    geom_smooth(method = "lm") +
    labs(y = "Quantity", x = "Gross Income", title = "Linear Regression: Gross Income vs. Quantity")
```

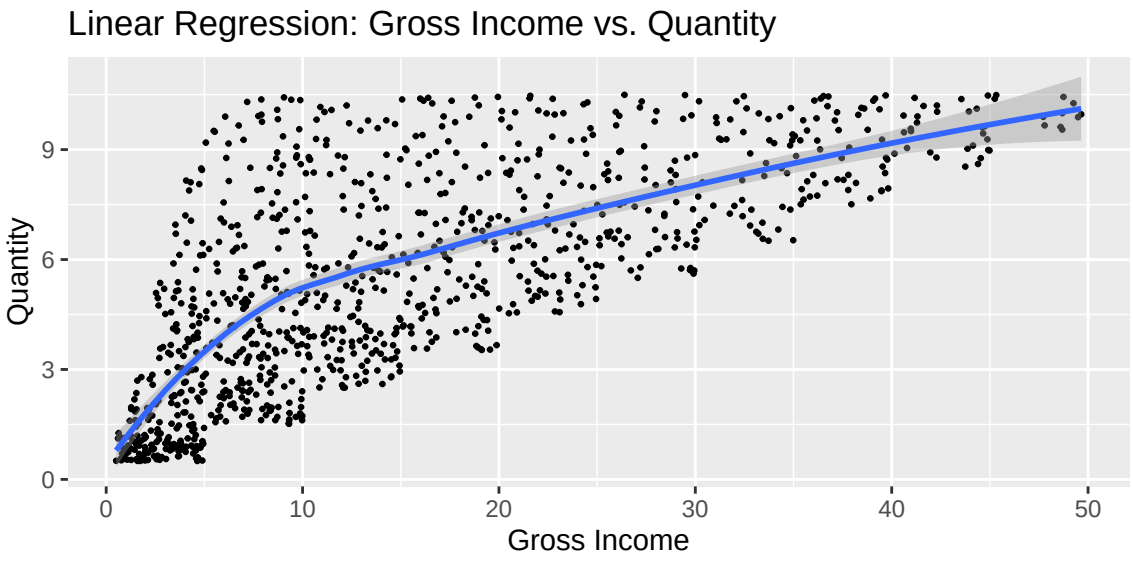
```
## `geom_smooth()` using formula = 'y ~ x'
```



In this plot, we use a polynomial curve instead of a straight line.

```
ggplot(sdata, aes(y = Quantity, x = gross.income)) +
    geom_point(position = position_jitter(width = 0, height = 0.5), size = 0.5) +
    geom_smooth(method = "loess") +
    labs(y = "Quantity", x = "Gross Income", title = "Linear Regression: Gross Income vs. Quantity")
```

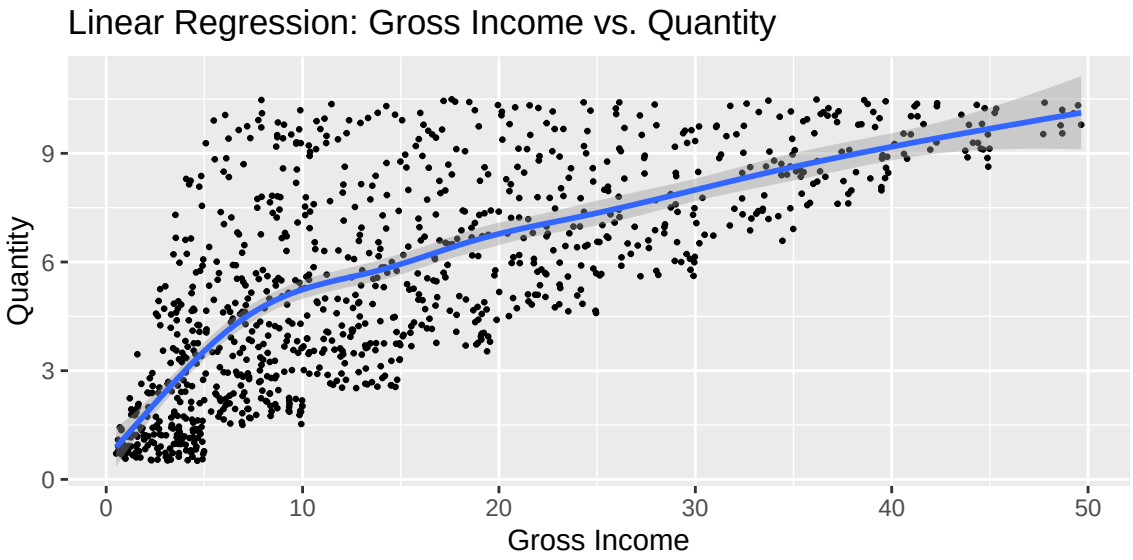
```
## `geom_smooth()` using formula = 'y ~ x'
```



If you don't specify any method of curve, it will assume automatically.

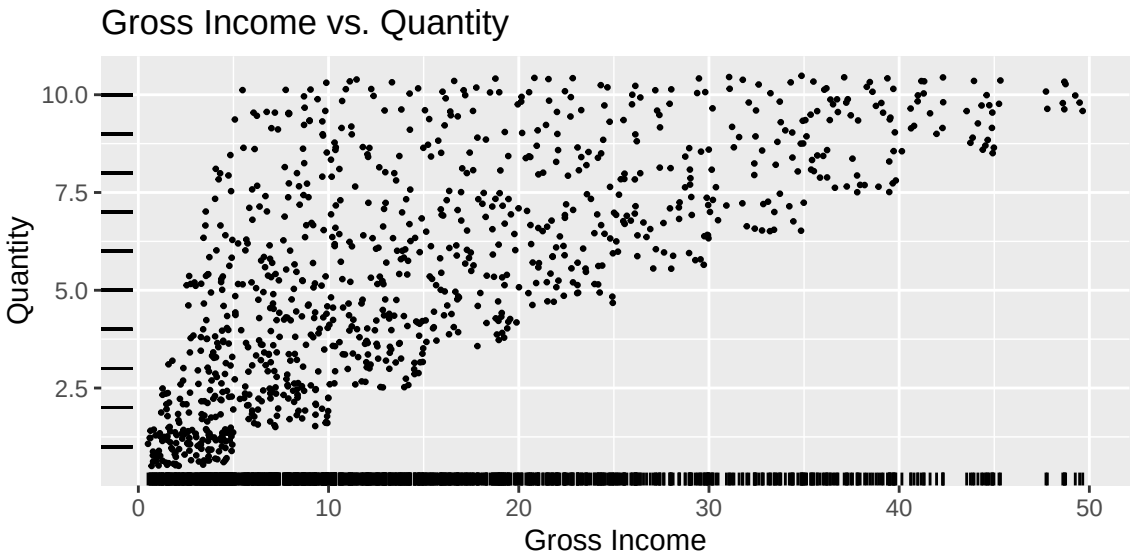
```
ggplot(sdata, aes(y = Quantity, x = gross.income)) +
    geom_point(position = position_jitter(width = 0, height = 0.5), size = 0.5) +
    geom_smooth() +
    labs(y = "Quantity", x = "Gross Income", title = "Linear Regression: Gross Income vs. Quantity")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



We can add marginal rugs to a scatter plot using geom_rug().

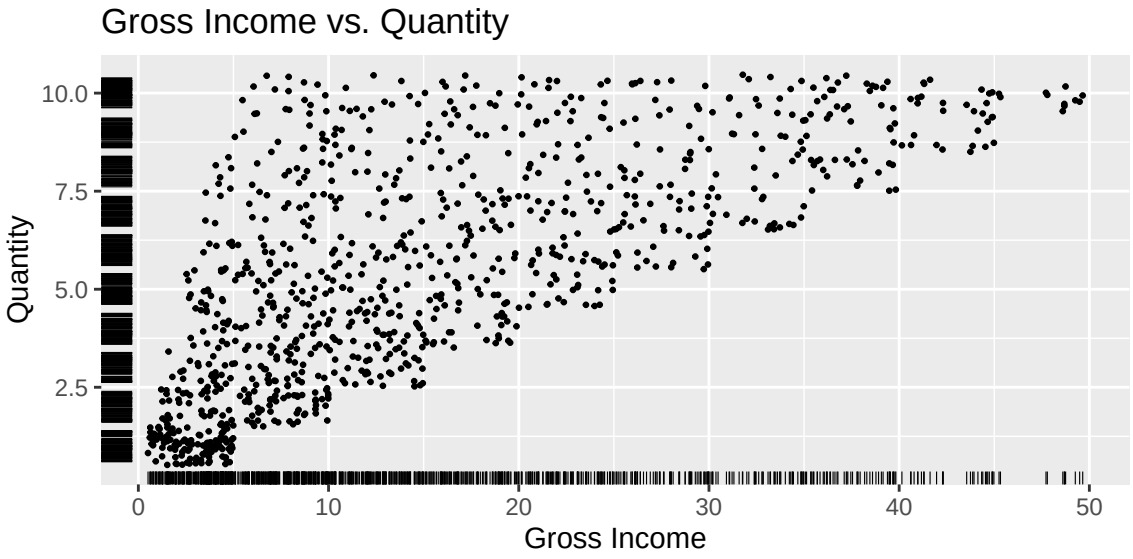
```
ggplot(sdata, aes(y = Quantity, x = gross.income)) +
    geom_point(position = position_jitter(width = 0, height = 0.5), size = 0.5) +
    labs(y = "Quantity", x = "Gross Income", title = "Gross Income vs. Quantity") +
    geom_rug()
```



We can add a marginal rug with jitter.

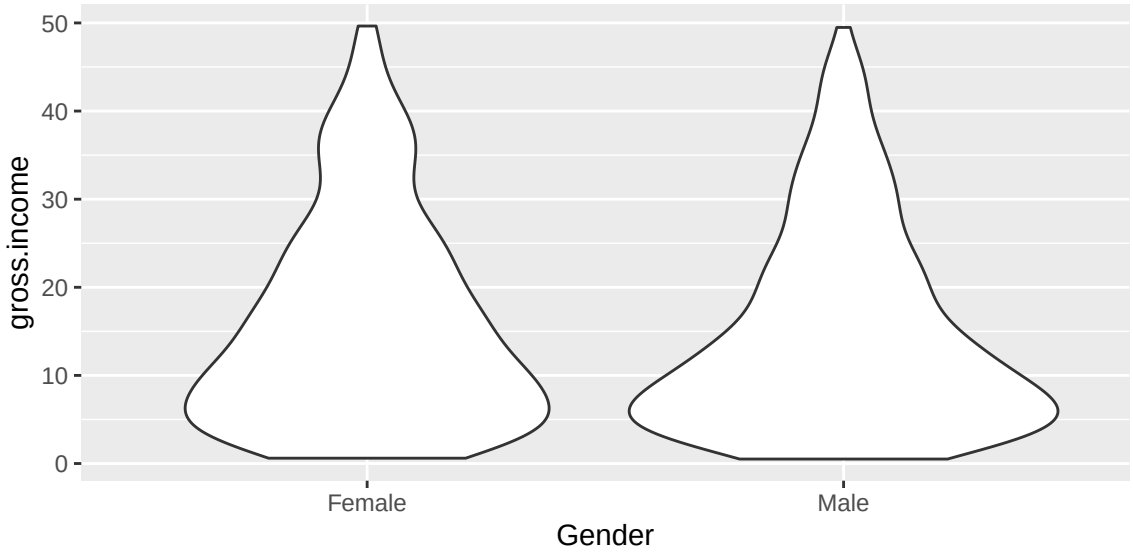
```
ggplot(sdata, aes(y = Quantity, x = gross.income)) +
    geom_point(position = position_jitter(width = 0, height = 0.5), size = 0.5) +
    labs(y = "Quantity", x = "Gross Income", title = "Gross Income vs. Quantity") +
```

```
geom_ru
```



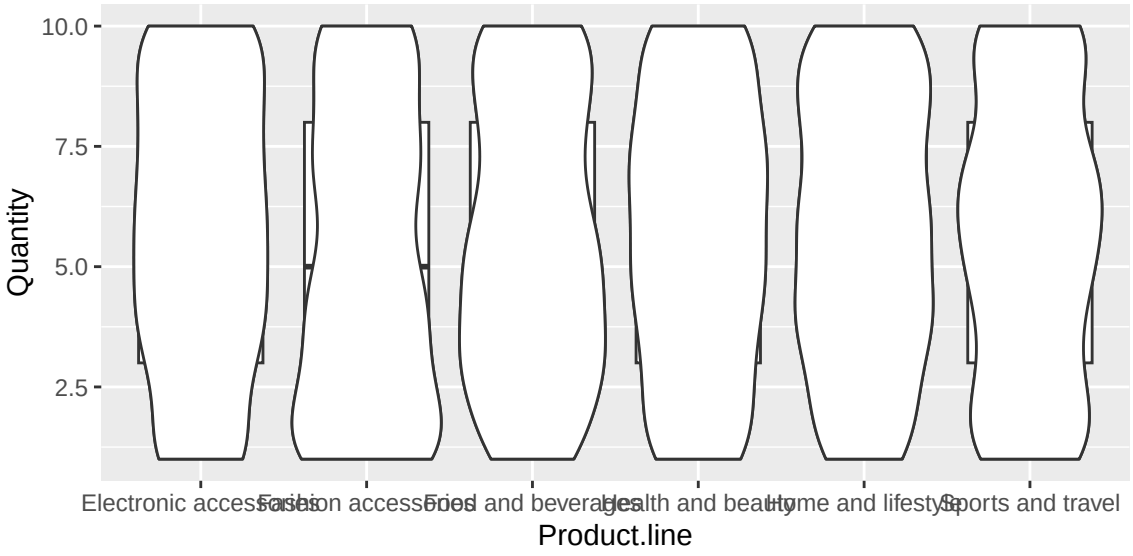
We can produce a violin plot using geom_violin().

```
ggplot(sdata, aes(x=Gender, y = gross.income)) + geom_violin()
```



We can combine Violin plot and box plot.

```
ggplot(sdata, aes(x=Product.line, y = Quantity)) + geom_violin() +                    geom_boxpl
  geom_violin()
```



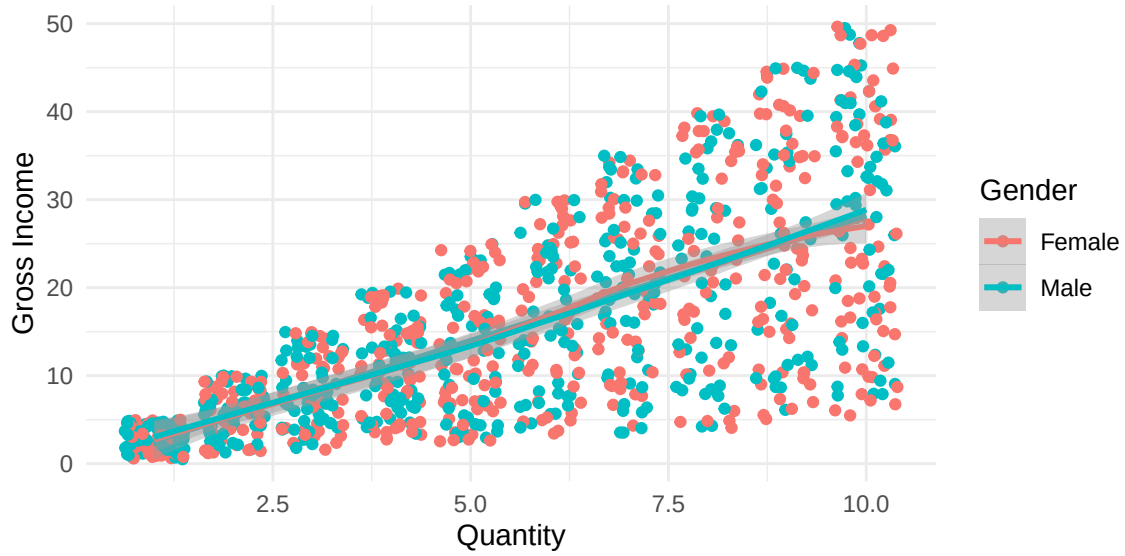
We can also plot many linear regression lines.

```
library(dplyr)
library(ggplot2)

# Create linear regression models
models <- sdata %>%
    group_by(Gender) %>%
```

```
    do(model = lm(gross.income ~ Quantity, data = .)) %>%
    ungroup()

# Generate predicted values
predvals <- models %>%
    group_by(Gender) %>%
    do(data.frame(Quantity = seq(min(sdata$Quantity), max(sdata$Quantity), length.out = 100),
                  Predicted = predict(.$model[[1]], newdata = data.frame(Quantity = seq(min(sdata$Quant
    ungroup()

# Plot scatter plot and predicted lines
ggplot(sdata, aes(x = Quantity, y = gross.income, colour = Gender)) +
    geom_point(position = "jitter") +
    geom_line(data = predvals, aes(x = Quantity, y = Predicted), size = 1) +
    labs(x = "Quantity", y = "Gross Income") +
    theme_minimal() + geom_smooth()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

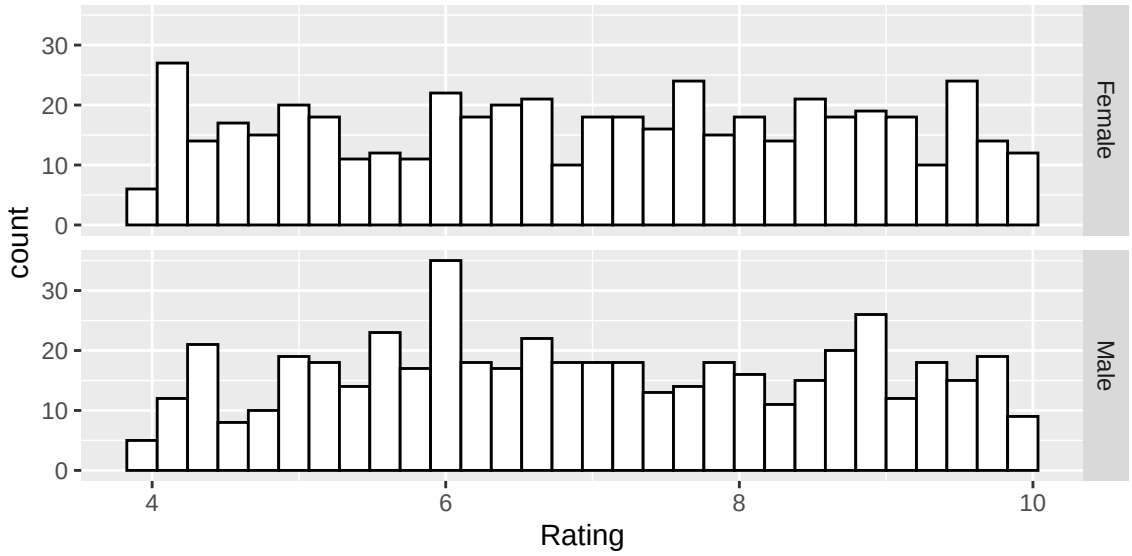
```
## ‘geom_smooth()‘ using method = ’loess’ and formula = ’y ~ x’
```



We can use facet variable and create different histograms.

```
ggplot(sdata, aes(x = Rating)) + geom_histogram(fill="white", color="black") + facet_grid(Gender ~ .)
```

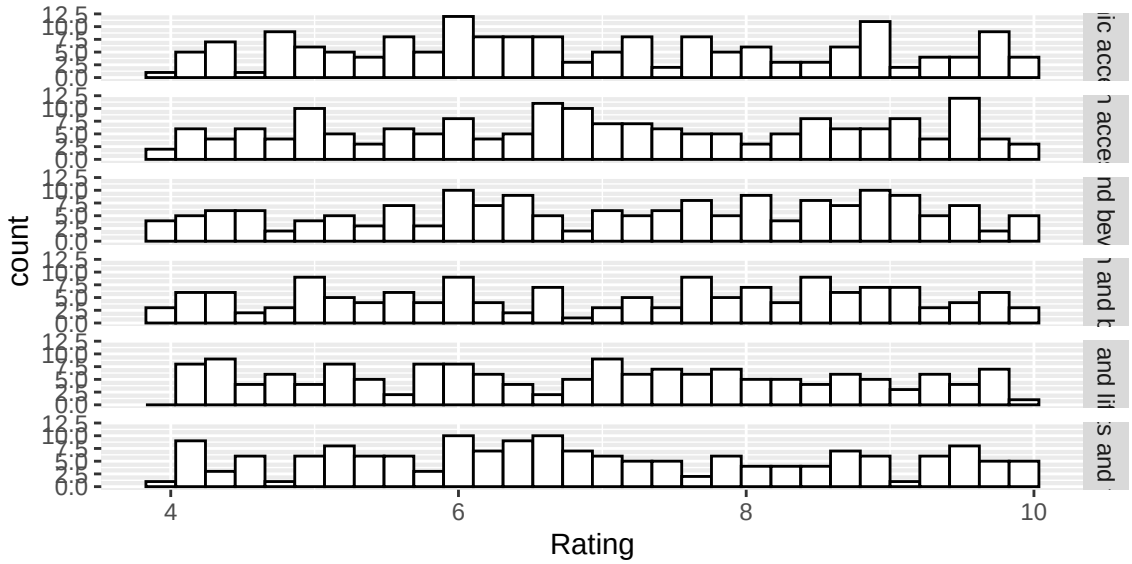
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



Another example with a different facet variable.

```
ggplot(sdata, aes(x = Rating)) + geom_histogram(fill="white", color="black") + facet_grid(Product.line
```

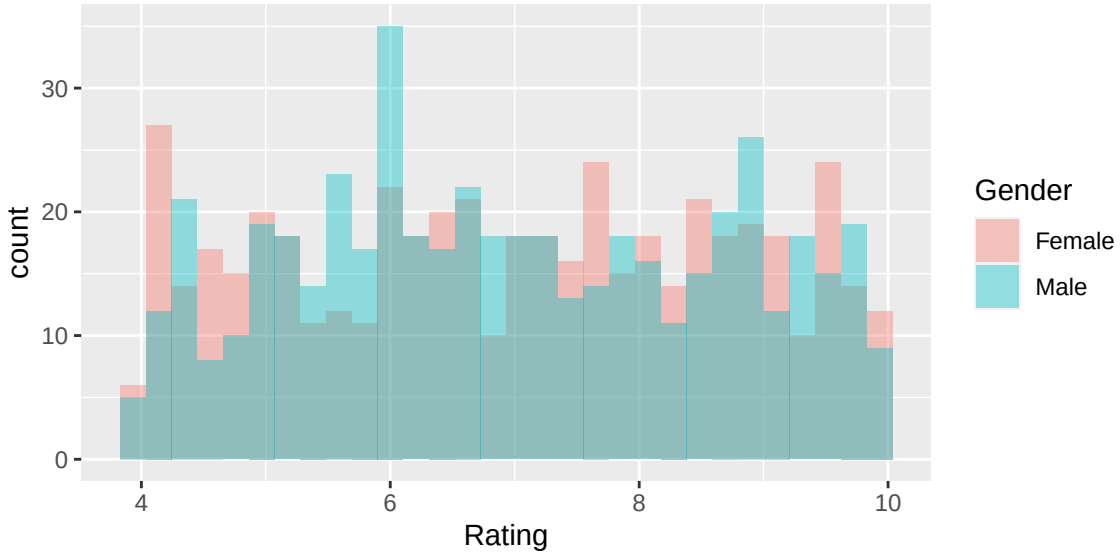
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can use the fill parameter to show two in one histogram.

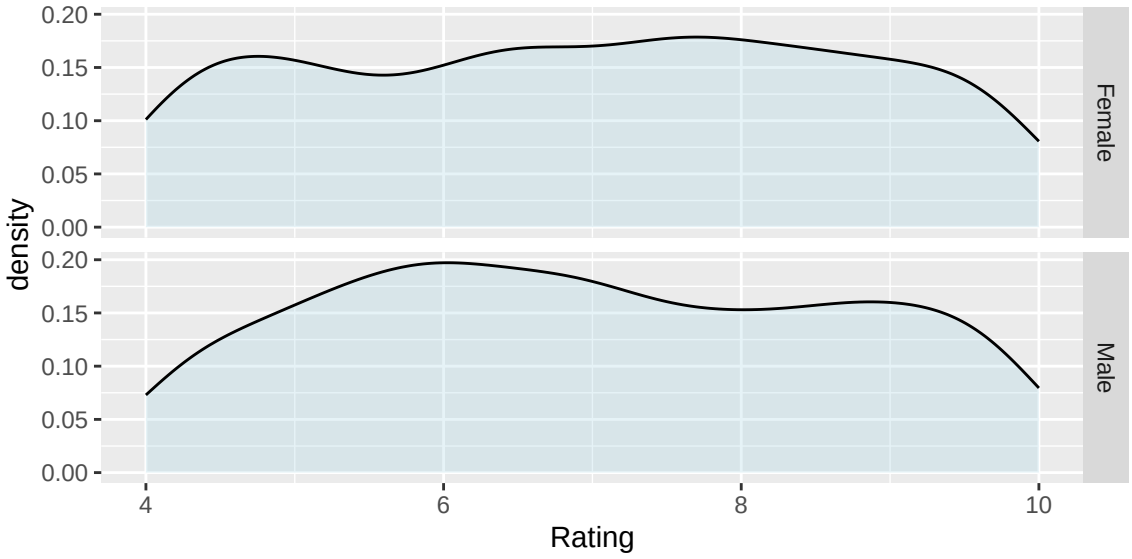
```
ggplot(sdata, aes(x = Rating, fill=Gender)) + geom_histogram(position = "identity", alpha = 0.4)
```

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



We can plot density curves with facet variables.

```
ggplot(sdata, aes(x = Rating))+ geom_density(alpha = .3, fill="lightblue") + facet_grid(Gender ~ .)
```

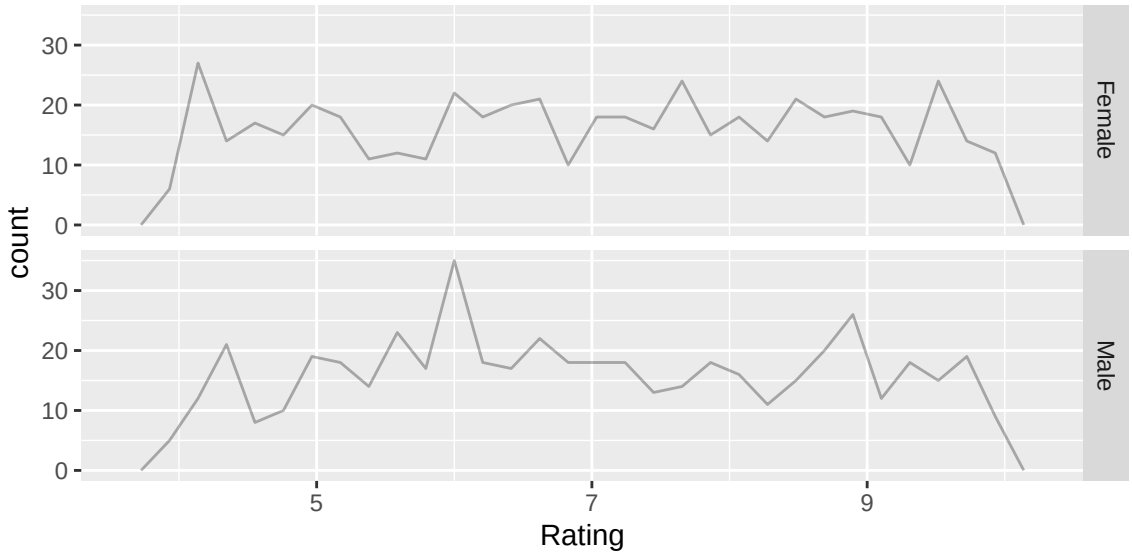


We can plot Frequency curves with facet variables.

```
ggplot(sdata, aes(x = Rating))+ geom_freqpoly(alpha = .3, fill="lightblue") + facet_grid(Gender ~ .)
```

```
## Warning in geom_freqpoly(alpha = 0.3, fill = "lightblue"): Ignoring unknown
## parameters: `fill`
```

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


count
30
20
10
0
Female
30
20
10
0
Male
5
7
9
Rating