CS 251 Statistical Computing

HOP 3: R for statistical project

4/12/2020 Developed by Aya Khalil

5/9/2020 Reviewed by Amrutha

School of Technology and Computing @City University of Seattle (CityU)

**Before You Start**

* If you already finished this module through any CityU Technology Institute (TI) courses,  
  just skim this module and skip it.
* Version numbers may not match with the guide. But that should be fine.  
  If given the option to choose between stable release (long-term support) or most recent, please choose the stable release.
* This guide targets Windows OS users. So, MacOS users may have different commands to input in the shell/terminal.
* We cannot explain every step. **This cookbook always needs your own creative judgement.**
* **For your working directory, use your course number.** The hands-on tutorial may use a different course number as an example.

**Learning Outcomes**

* Data Visualization

**Resource**

* Hui, E. G. M. (2019). [*Learn R for applied statistics: With data visualization, regressions, and statistics*](https://login.proxy.cityu.edu/sso/skillport?context=144516). Apress.
* Data Science and Machine Learning BootCamp with R online course

**Introduction to Data Visualization**

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

Data visualization can involve the plotting of a bar, histogram, scatterplot, boxplot, line chart, time series, and scatterplot matrix chart to help us analyze the data and understand the causality and relationship between variables. Data visualization can also be viewed as descriptive statistics to some. Humans distinguish differences in line, shape, and color without much processing effort, and data visualization can take advantage of this to create charts and graphs to help users understand the data more easily

As we go through this tutorial on ggplot2, it may be helpful to use this useful cheat sheet for reference when using ggplot2 <https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>

**Layers for building Visualizations**

ggplot2 is based off the grammar of graphics, which sets a paradigm for data visualization in layers:



We won't go too much in depth to all philosophy of the grammar of graphics because the best source of this is from the creator of ggplot, Hadley Wickham, who created a great paper on the topic which you can read here: <http://vita.had.co.nz/papers/layered-grammar.pdf>.

The general syntax of using ggplot2 will look like this:

ggplot(data = <default data set>,

aes(x = <default x axis variable>,

y = <default y axis variable>,

... <other default aesthetic mappings>),

... <other plot defaults>) +

geom\_<geom type>(aes(size = <size variable for this geom>,

... <other aesthetic mappings>),

data = <data for this point geom>,

stat = <statistic string or function>,

position = <position string or function>,

color = <"fixed color specification">,

<other arguments, possibly passed to the \_stat\_ function) +

scale\_<aesthetic>\_<type>(name = <"scale label">,

breaks = <where to put tick marks>,

labels = <labels for tick marks>,

... <other options for the scale>) +

theme(plot.background = element\_rect(fill = "gray"),

... <other theme elements>)

**Please watch the following video to understand the concept of data visualization**

*“ggplot2 Tutorial | ggplot2 In R Tutorial | Data Visualization In R | R Training | Edureka”*

At <https://www.youtube.com/watch?v=N5gYo43oLE8>

**Setup Working Environment for Module3**

1. Open VS Code.

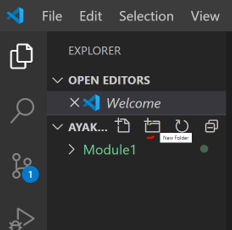
* **online student:** Open CS251 \_Fall\_2020/**ON**/FirstnameLastname /. ( File > Open )
* **onsite student:** Open CS251 \_ Fall \_2020/**IN**/FirstnameLastname. ( File > Open )

1. Then, create the “**Module3**” directory in the VSCode.

>>>mkdir Module3

OR

Click on New Folder button that behind your name, and name the folder Module3



* **In Module3** project folder, create new file DVHist.R

Open the R console

-Type the following command in the R console to install ggplot2

>install.packages(‘ggplot2’,repos = 'http://cran.us.r-project.org')

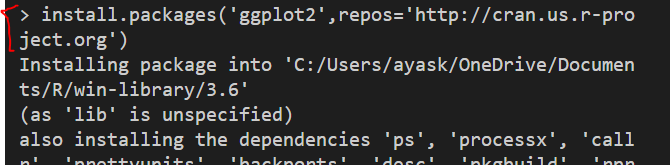
***Note: If the installation command did not work, try***

>install.packages(‘ggplot2’)

Then chose any of the USA CRAN

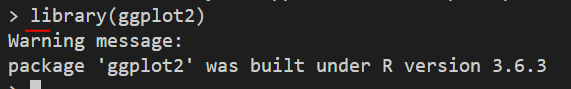
***If the installation command did not work, try***

>install.packages(“ggplot2”)



-Type the following in the R console

>library(ggplot2)



# Histograms with ggplot2

Histograms are a great way of quickly exploring your data! We have a couple of options for quickly producing histograms off the columns of a data frame. We have:

* hist()
* qplot()
* ggplot()

They differ mainly in one main component, for each of these methods you usually trade-off ease of use for ability to customize.

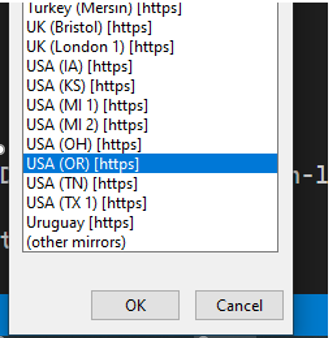
**Load Data**

We'll use the movie dataset that comes with ggplot:

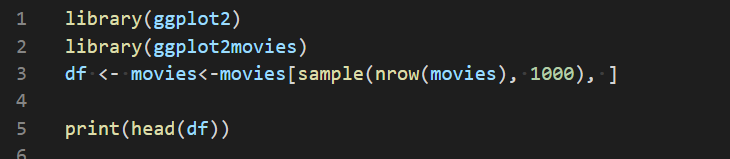
-In the R console, type the following to install the movies package

>install.packages(‘ggplot2movies’)

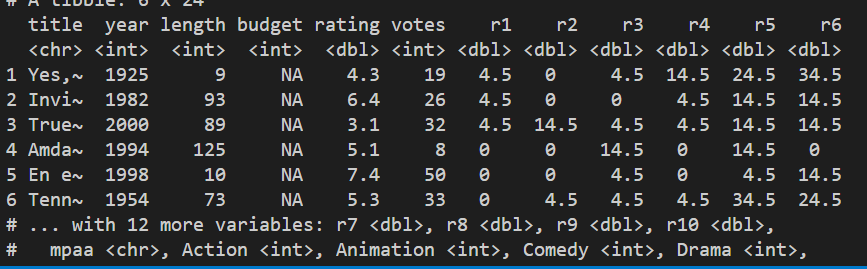
Choose any of the USA crans



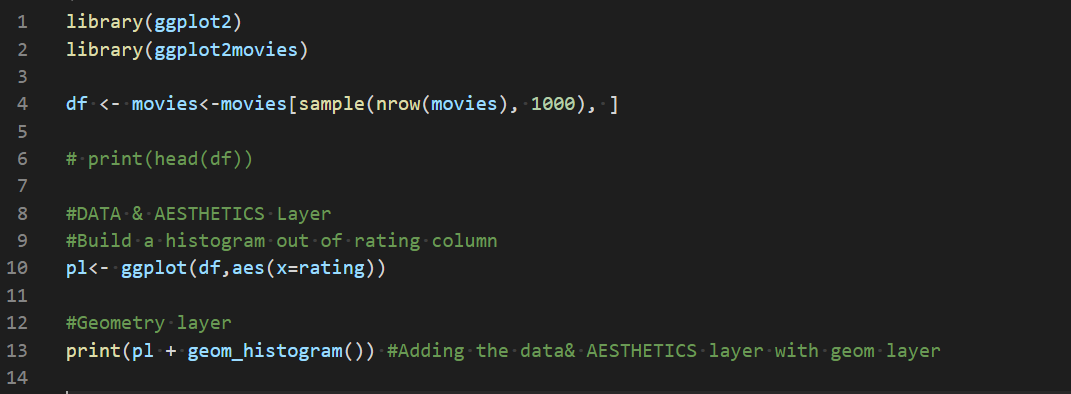
-Type the following code in DVHist.R file



-Save & Run your code

o/p: 

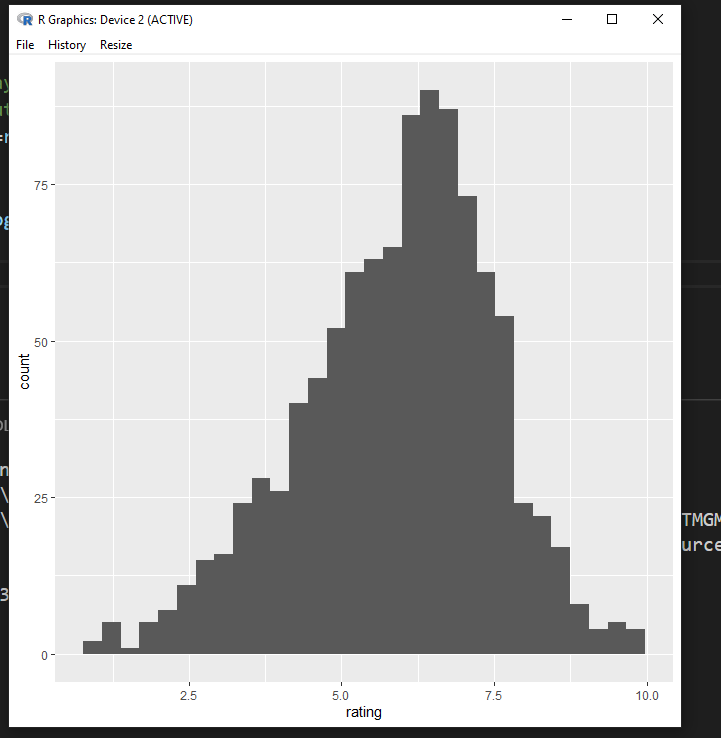
- Add the following code to DVHist.R file



- The x-axis is the rating columns

-Save & Run your code

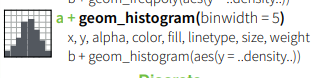
o/p



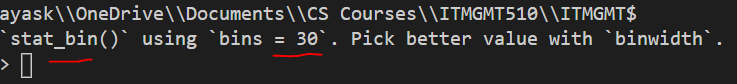
This is the very basic of the histogram looks like.

-We can add arguments in the Geometry layer in order to customize our plot

-If you go to the [cheat sheet](https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf), you will find the arguments that we can add to the geometry layer

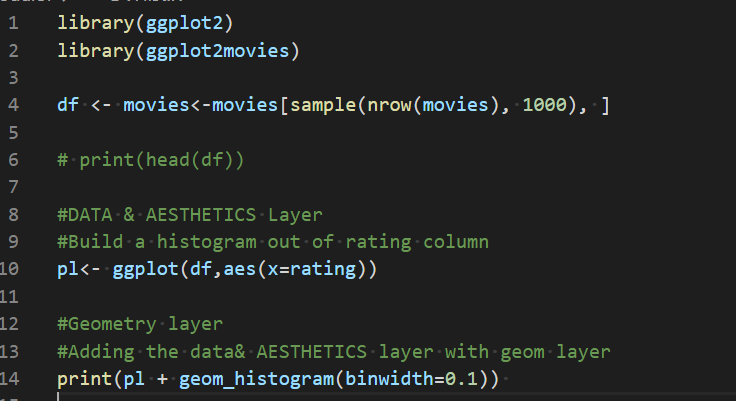


If you take a look in the o/p in the R console you will find



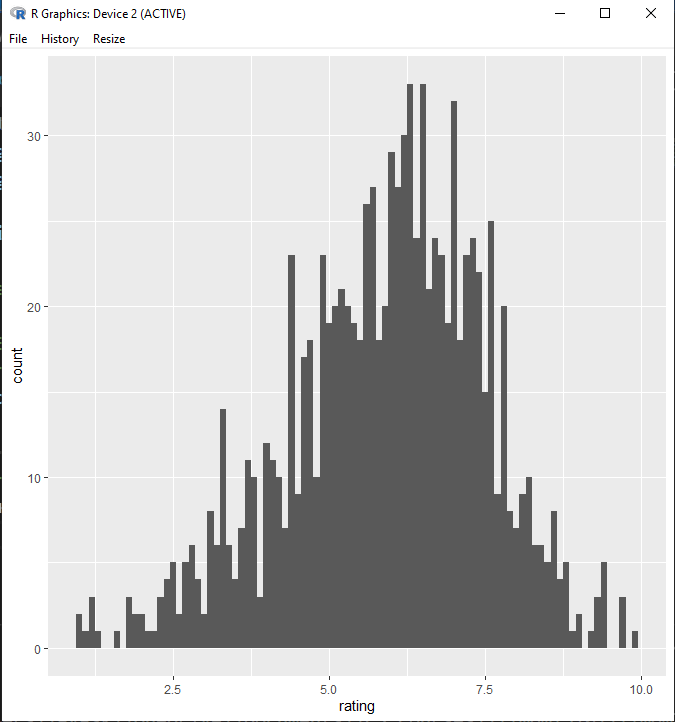
So, lets add the argument binwidth and see the o/p

- Add the following code to DVHist.R file



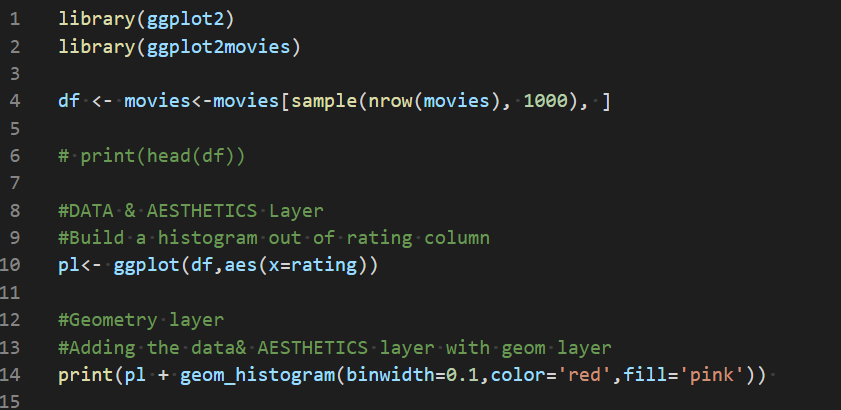
-Save & Run your code

-o/p

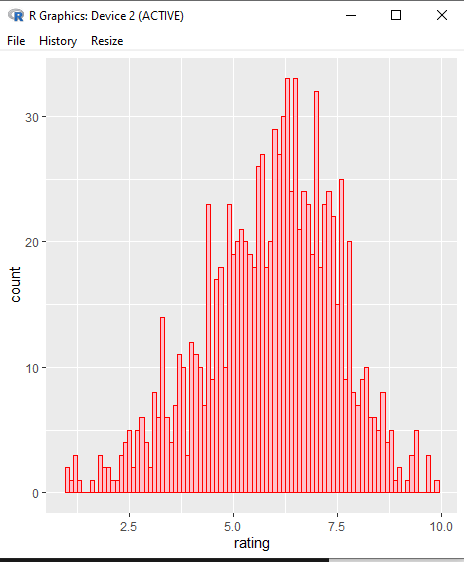


Also, we can change the color of the graph

- Add the following code to DVHist.R file

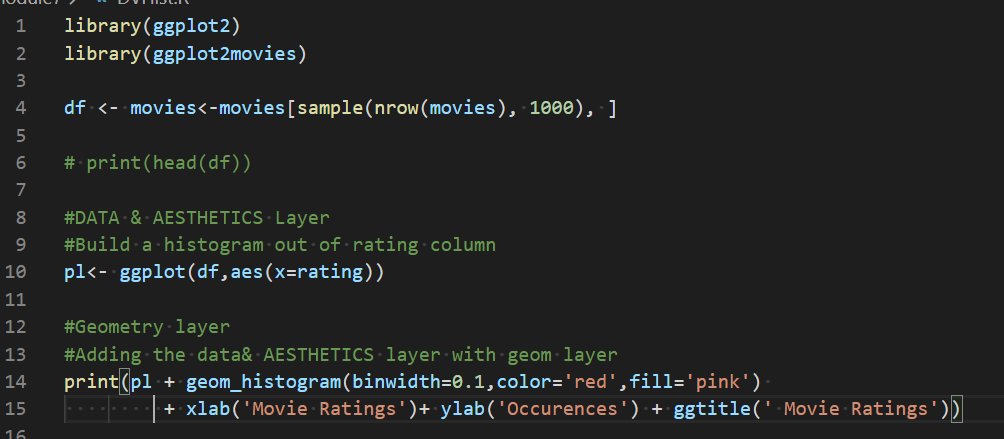


-Save &Run

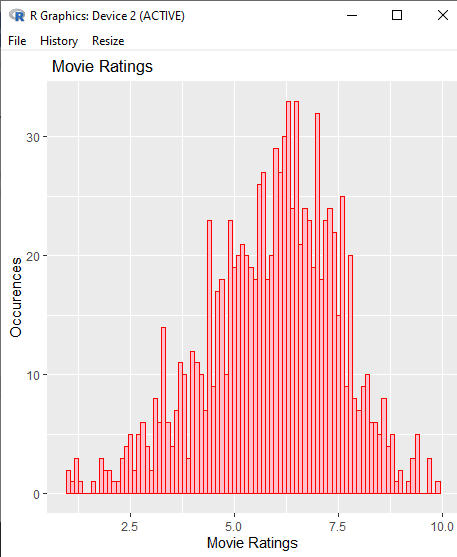
o/p 

**-You can add labels to your graph**

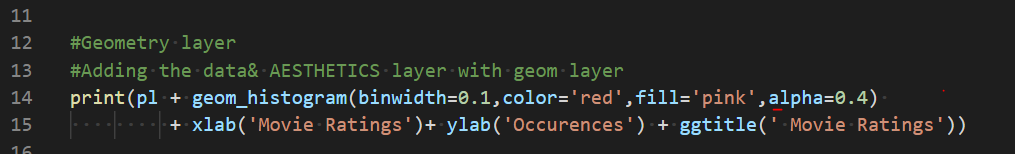
**-**Add the following to DVHist.R file



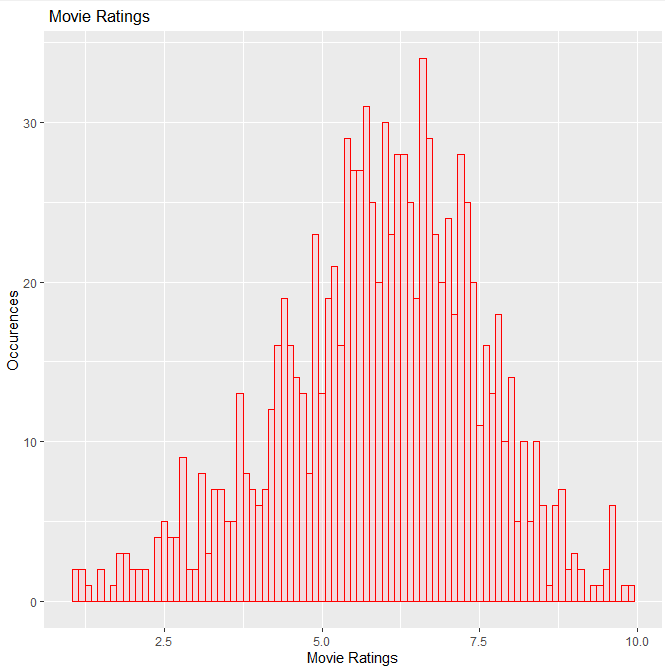
-Save & Run

o/p 

## Change Alpha (Transparency)

- Add alpha=0.4 to your previous code

-Run & Save your code

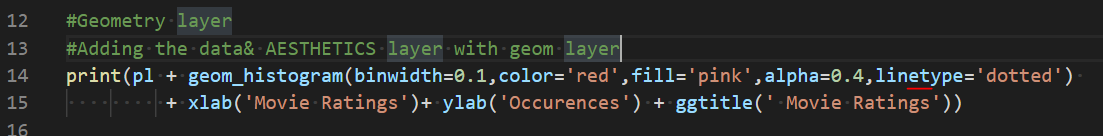
-o/p

-If alpha is equal to zero, then there will be no fill

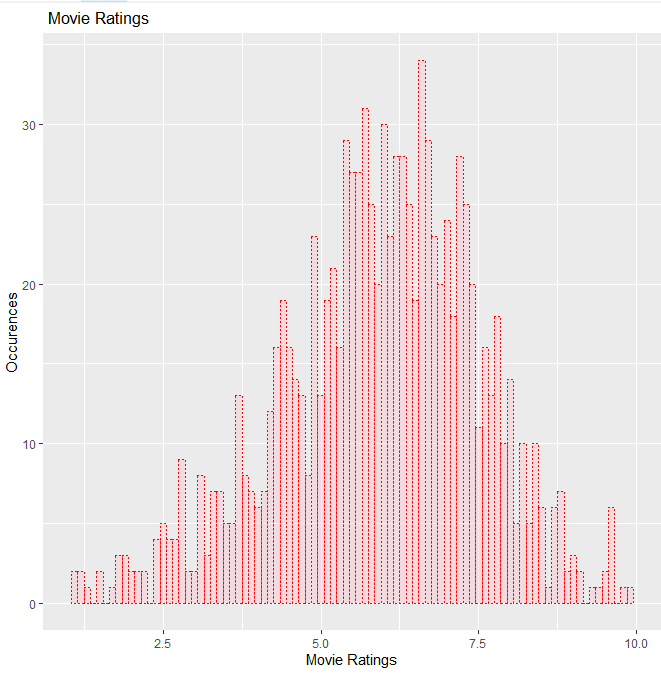
**Linetypes**

We have the options: "blank", "solid", "dashed", "dotted", "dotdash", "longdash", and "twodash". You would never really use these with a histogram, but just to show your options:

-Add linetypes to your previous code



-Save & Run your code

o/p 

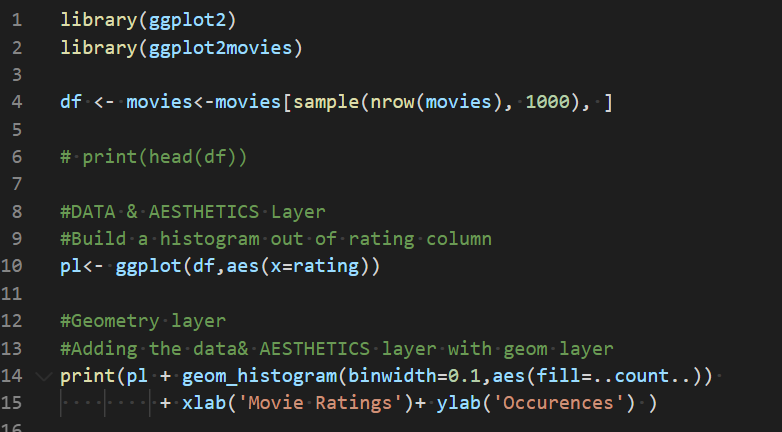
**Advanced Aesthetics**

We can add a aes() argument to the geom\_histogram for some more advanced features. We won't go too deep into these, but ggplot gives you the ability to edit color and fill scales.

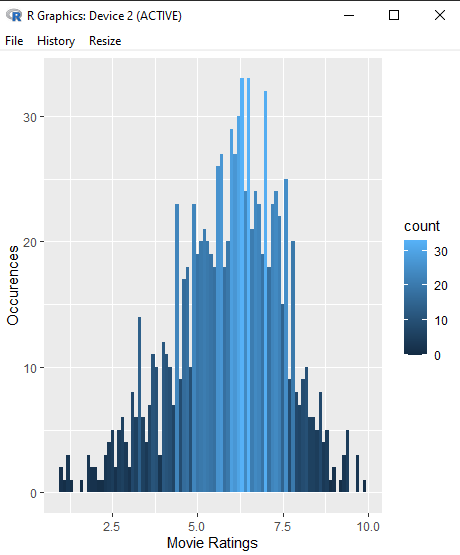
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DVHistAdvnc.R

Type the following in DVHistAdvnc.R



-Save & Run

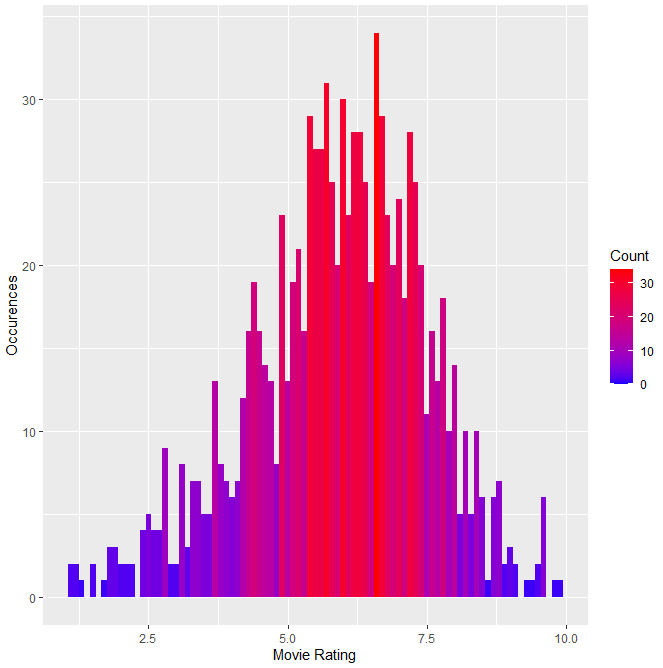


You can further edit this by adding the **scale\_fill\_gradient()** function to your ggplot objects:

-Add the following to DVHistAdvnc.R



-Save & Run

-o/p 

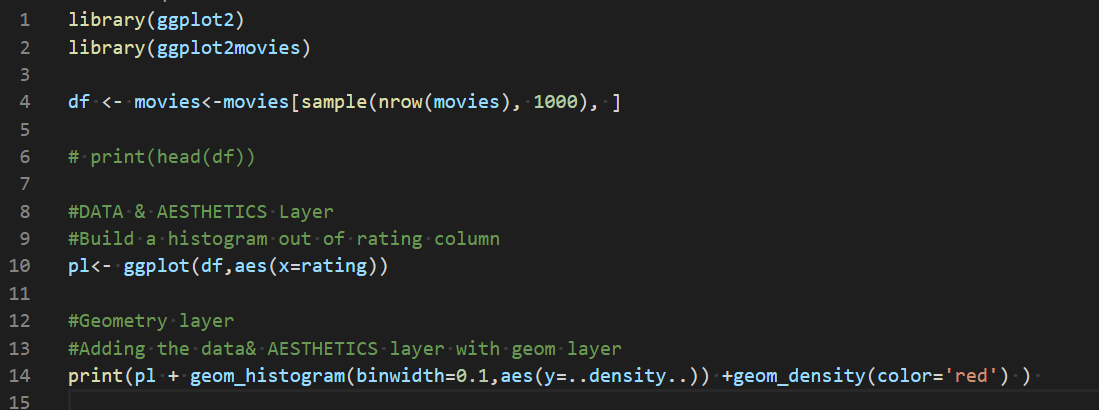
## **Adding density plot**

You can add a [kernel density estimation plot](https://en.wikipedia.org/wiki/Kernel_density_estimation)

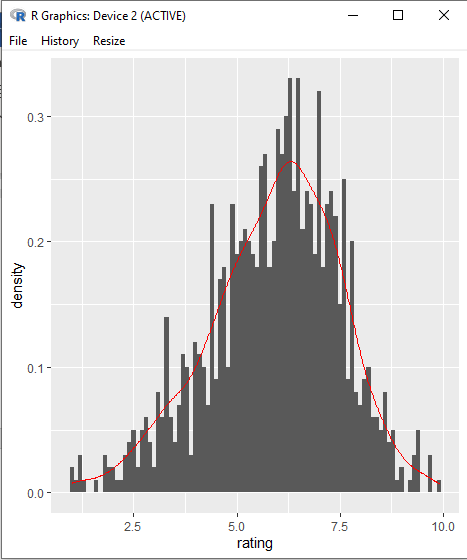
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DVdensityPlot.R

**Add the following to** DVdensityPlot.R



**Save& Run**

**o/p**

# Scatterplots with ggplot2

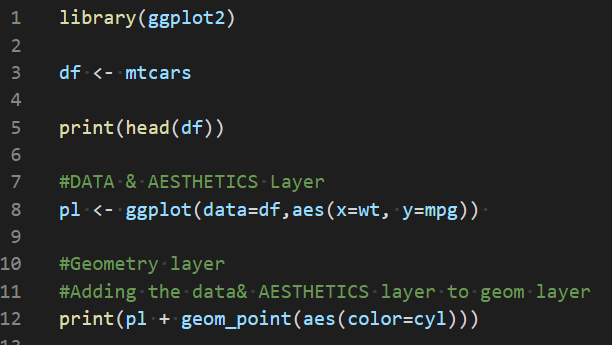
Scatter plots allow us to place points that let us see possible correlations between two features of a data set. Let's see how we can create them with ggplot!

We will use the built-in mtcars dataset:

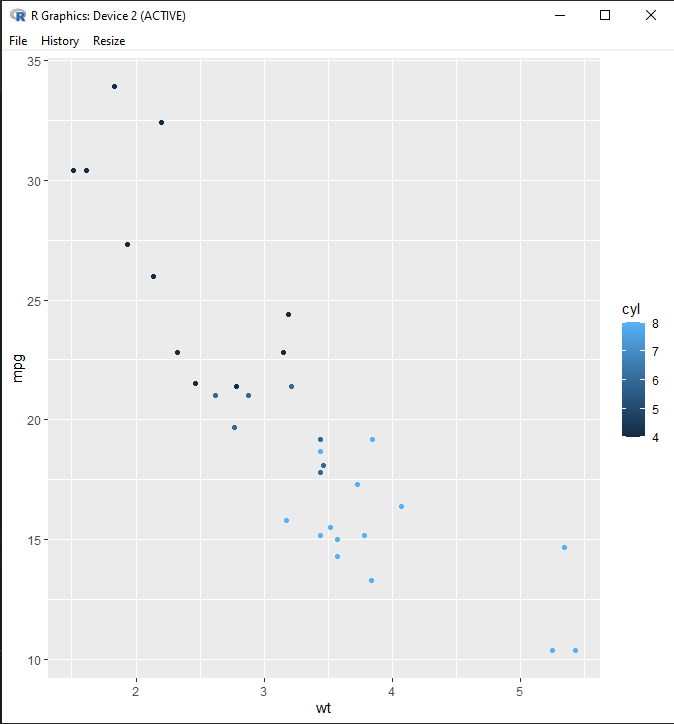
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-ScatterPlot.R

**Add the following to** DV-ScatterPlot.R



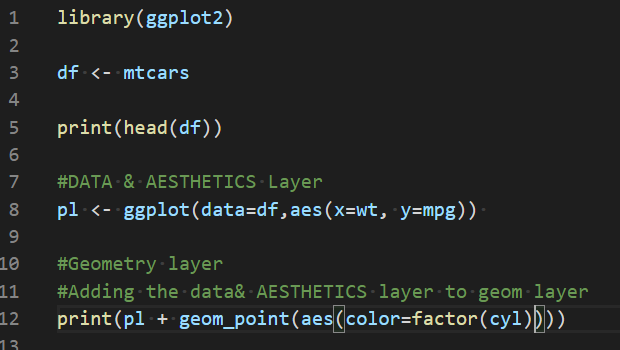
Save & Run

o/p 

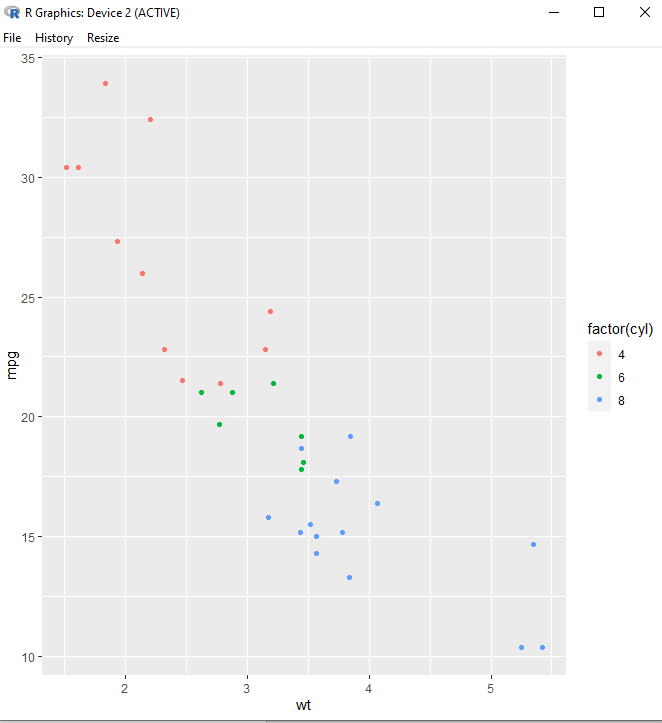
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** **project folder, create new file DV-ScatterPlot2.R**

**-**Add the following toDV-ScatterPlot2.R



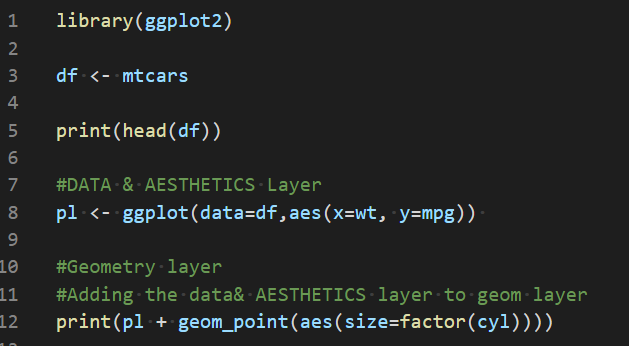
**-Save & Run**

**o/p** 

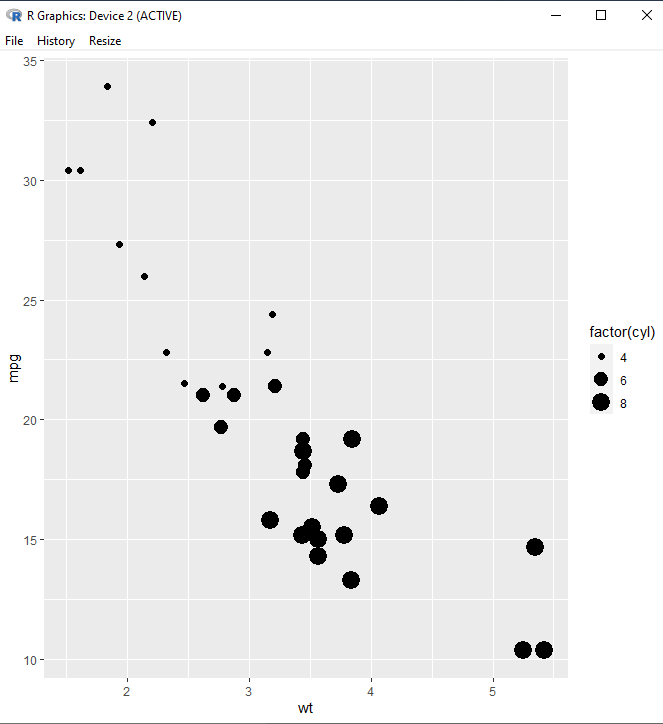
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-ScatterPlot3.R

**-**Add the following toDV-ScatterPlot3.R



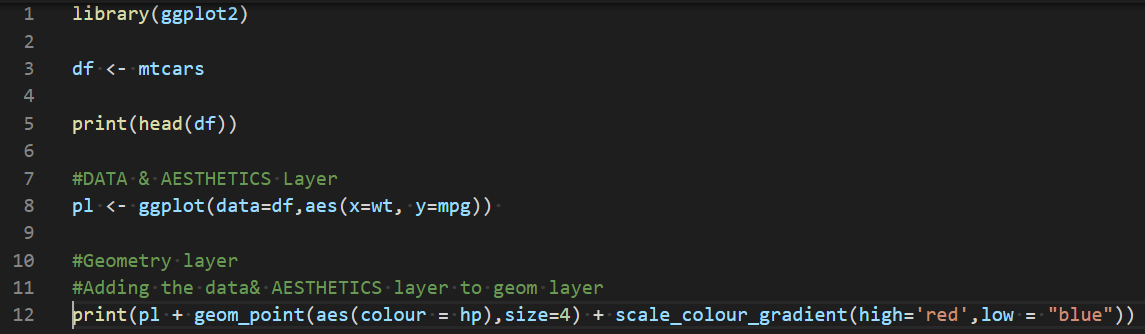
**-Save &Run**

**o/p**

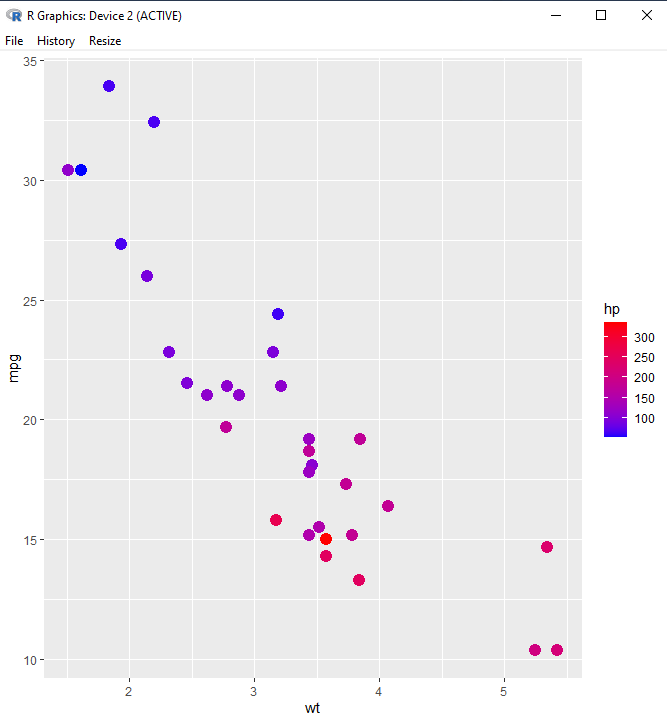
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-ScatterPlot4.R

Add the following toDV-ScatterPlot4.R



**-Save & Run**

**o/p** 

# Barplots with ggplot2

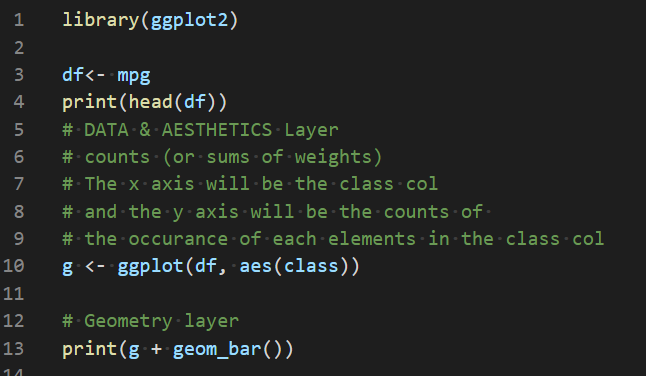
Barplots are a useful way of displaying occurrence counts when a histogram isn't quite what you're looking for! In ggplot2, there are two types of bar charts, determined by what is mapped to bar height. By default, geom\_bar uses stat="count" which makes the height of the bar proportion to the number of cases in each group (or if the weight aesthetic is supplied, the sum of the weights). If you want the heights of the bars to represent values in the data, use stat="identity" and map a variable to the y aesthetic.

Let's see how we can create them using the built-in mpg dataset.

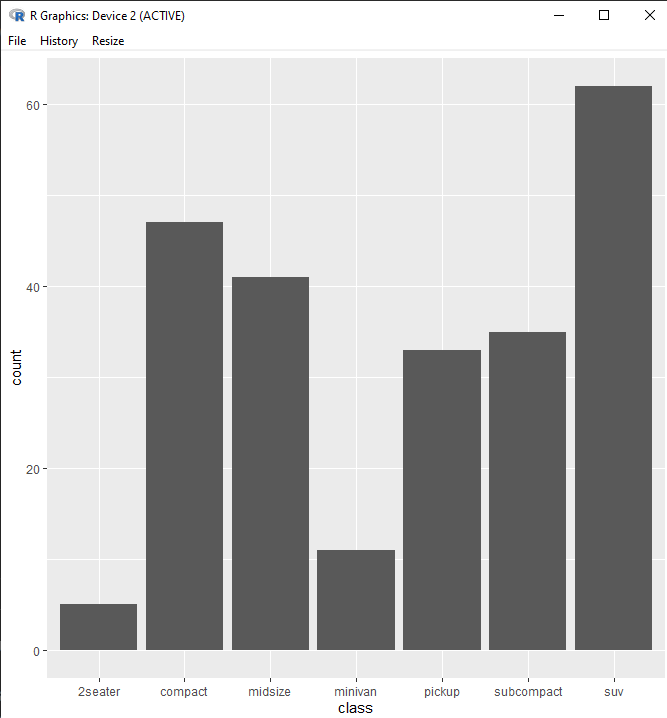
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-barPlot.R

**-**Add the following toDV- barPlot.R



-Save & Run

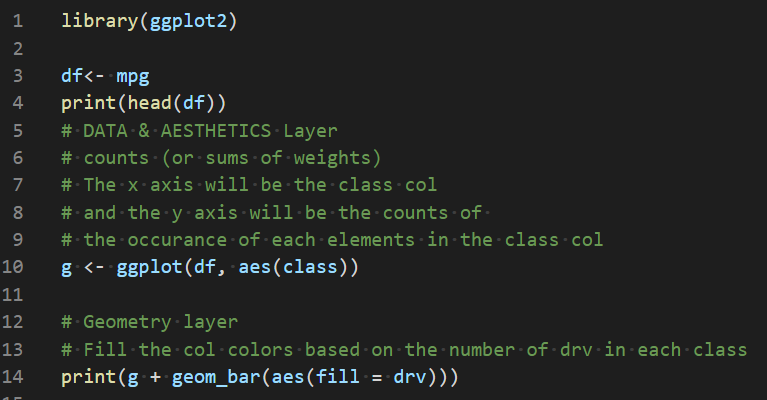
-O/P 

-Color the bar based on the drv columns

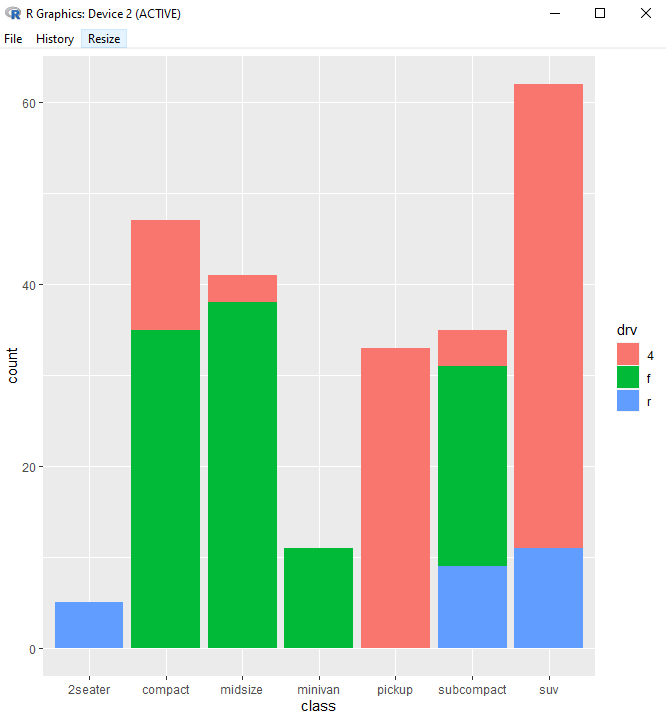
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-barPlot2.R

Add the following toDV- barPlot2.R



-Save & Run

o/p

# Boxplots with ggplot2

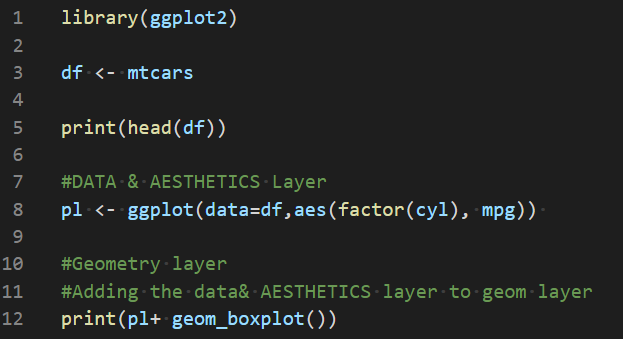
[Boxplots](https://en.wikipedia.org/wiki/Box_plot) are convenient way of graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. Outliers may be plotted as individual points.

Let's see how we can create them with ggplot

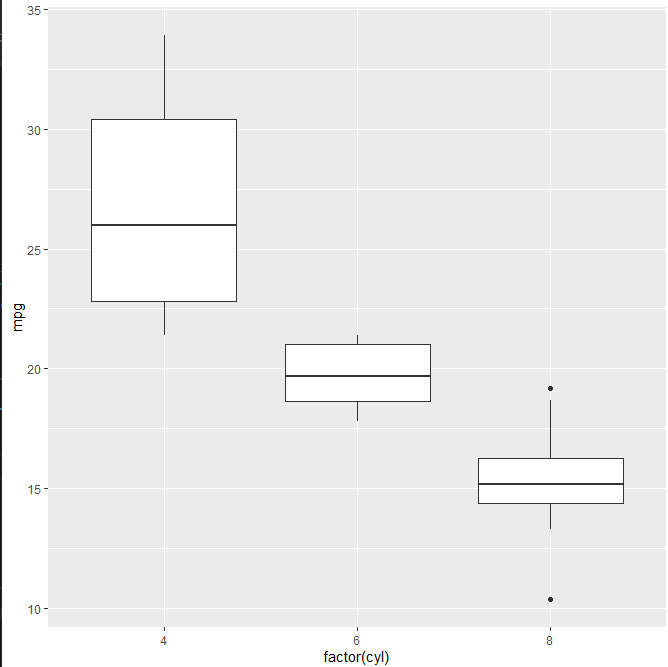
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-boxPlot.R

Add the following toDV- boxPlot.R

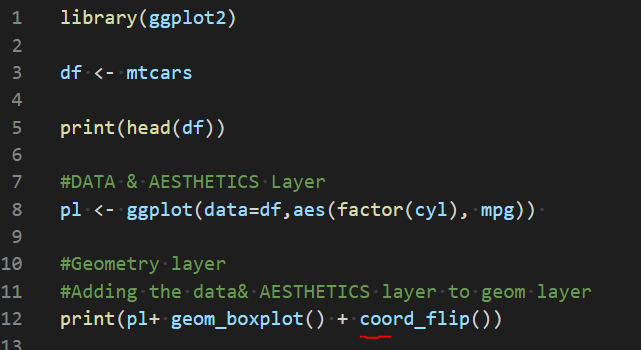


-Save & Run

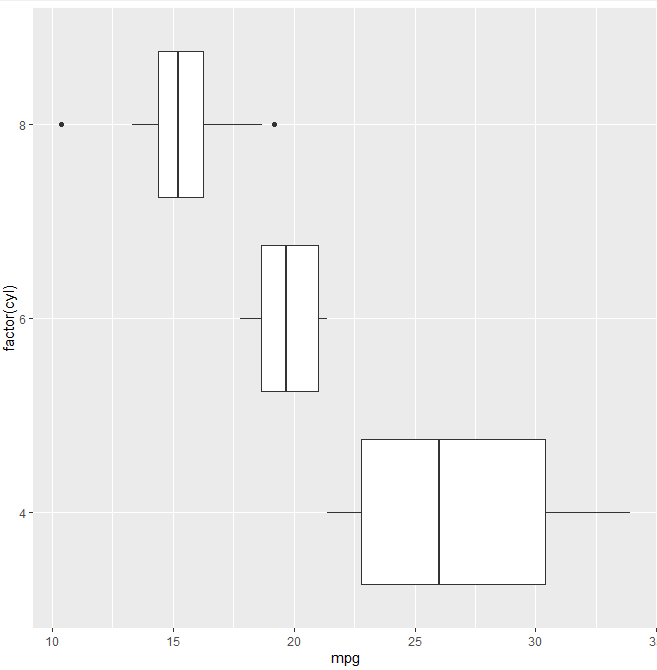


-We can make a coordination flip, which mean x-axis will be the y-axis

**-**Add the following toDV- boxPlot.R



-Save & Run



# Variable Plotting with ggplot2

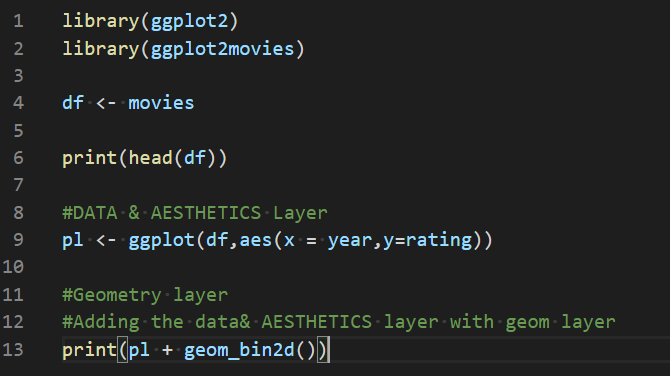
Ww will briefly show some examples of how you can compare two variables from a dataset. For these examples, we 'll need the full control of ggplot

**2d Bin Chart**

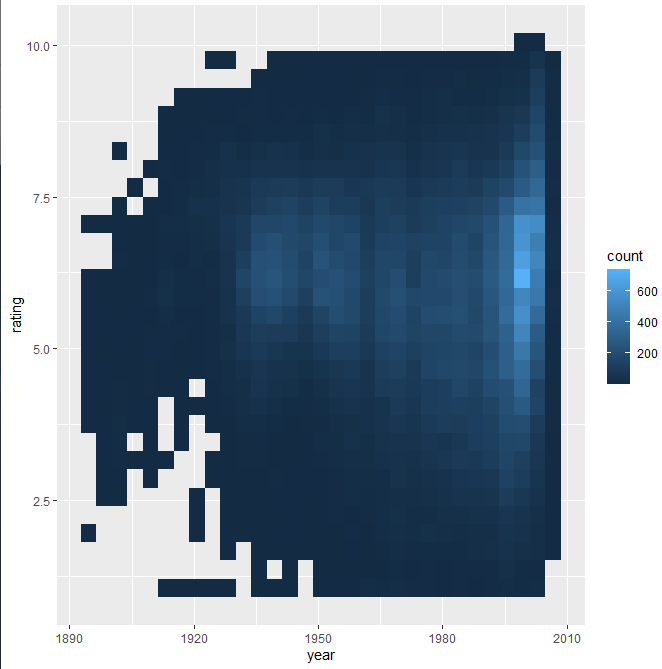
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3** project folder, create new file DV-VarPlot.R

Add the following toDV-VarPlot.R



-Save & Run

o/p 

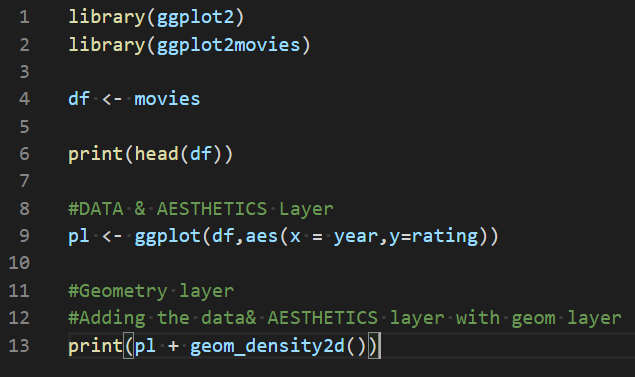
The output is relation between the years and the ratings, the colors express not the actual rating values but the number of occurrences.

**2d Density Plot**

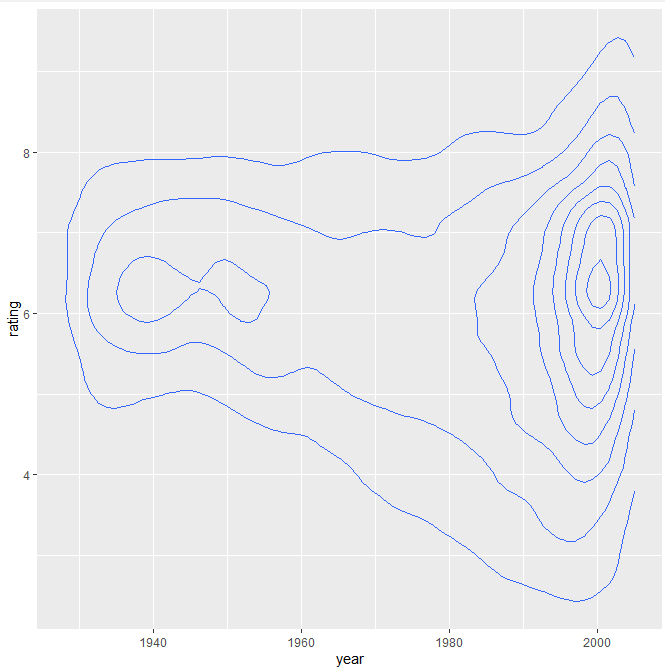
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* In Module3 project folder, create new file DV-VarPlot2.R

Add the following toDV-VarPlot2.R



Save & Run

o/p

**Coordinates and Faceting with ggplot2**

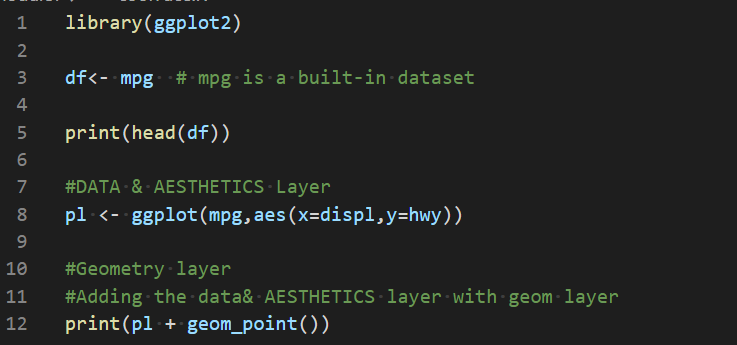
Learning how to deal with coordinates will allow us to size our plots correctly. Faceting will allow us to place several plots next to each other, these plots are usually related by the same dataset.

Let's build our understanding of these ideas by going through some simple examples

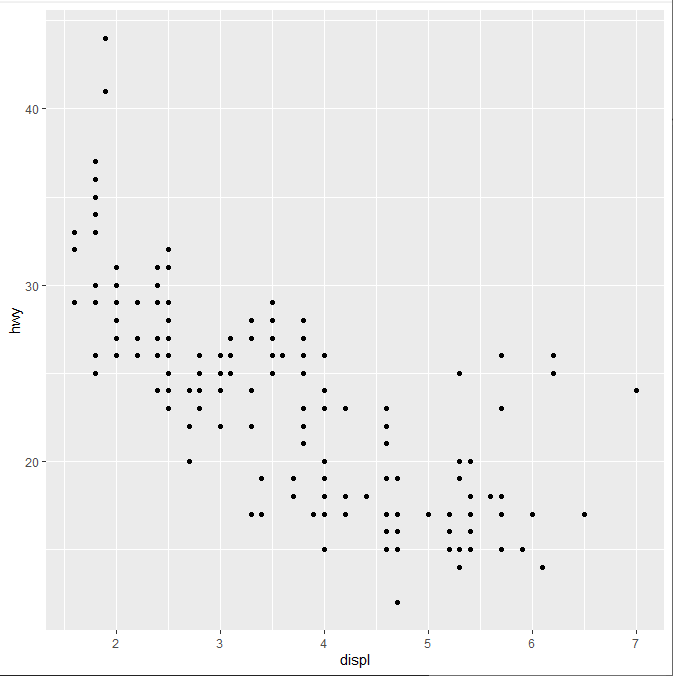
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3 project folder, create new file DV-CoorFace.R**

**Add the following to DV-CoorFace.R**



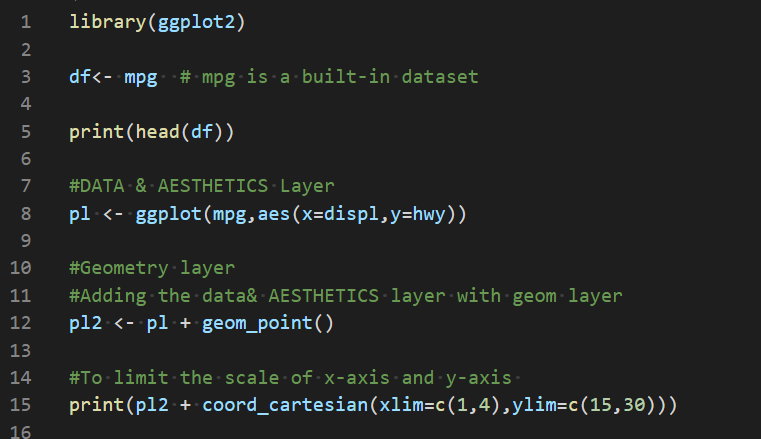
**-Save & Run**

**-O/P** 

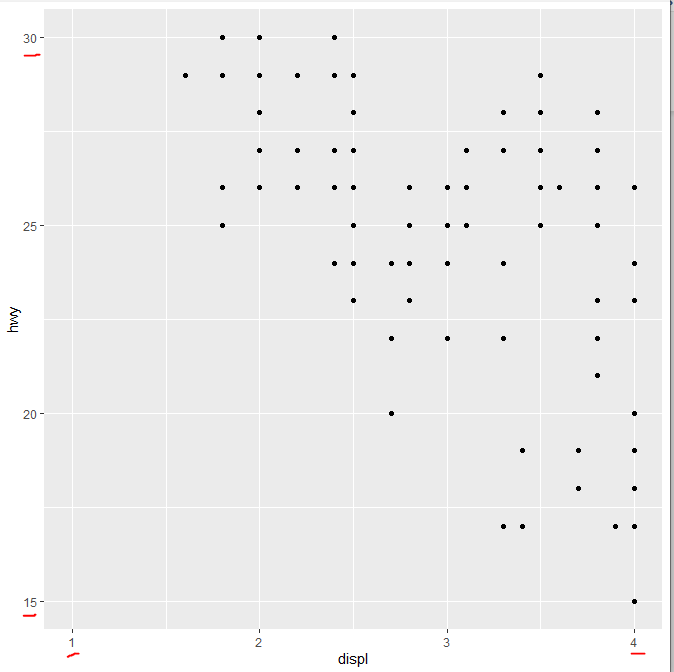
## **Setting x and y limits**

You can use **+ scale\_x\_continuous** and **scale\_y\_continuous** with an additional **limits=c(low,high)** argument to set the scale. A sometimes-nicer way to do this is by adding **+ coord\_cartesian()**with **xlim** and **ylim** arguments and pass in numeric vectors

* **Add the following to update DV-CoorFace.R**



**-Save & Run**



**-Notice that the limit of x-axis is 1 to 4 & y-axis is 15 to 30**

**Facets**

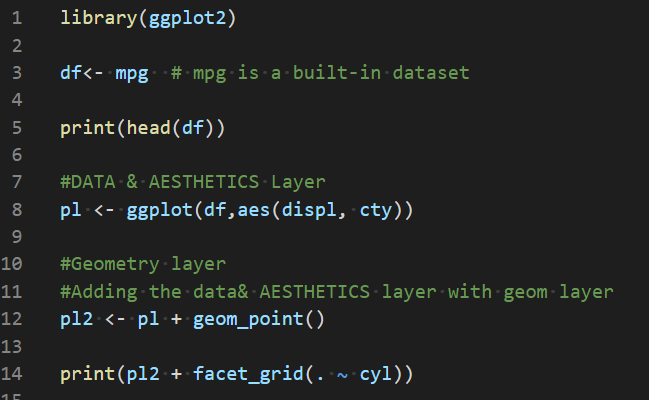
The best way to set up a facet grid (multiple plots) is to use facet\_grid(), the main part of this function is the use of the facet argument:

A formula with the rows (of the tabular display) on the LHS and the columns (of the tabular display) on the RHS; the dot in the formula is used to indicate there should be no faceting on this dimension (either row or column). The formula can also be provided as a string instead of a classical formula object.

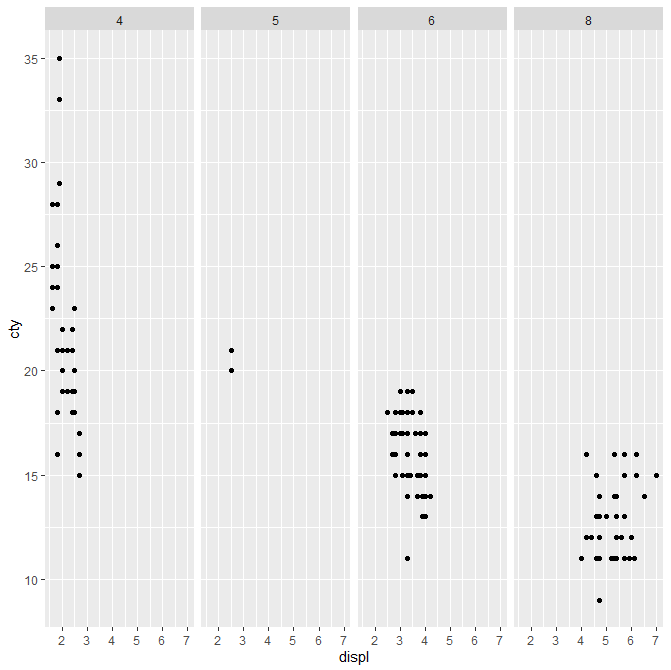
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3 project folder, create new file DV-FaceGrid.R**

**Add the following to DV-FaceGrid.R**



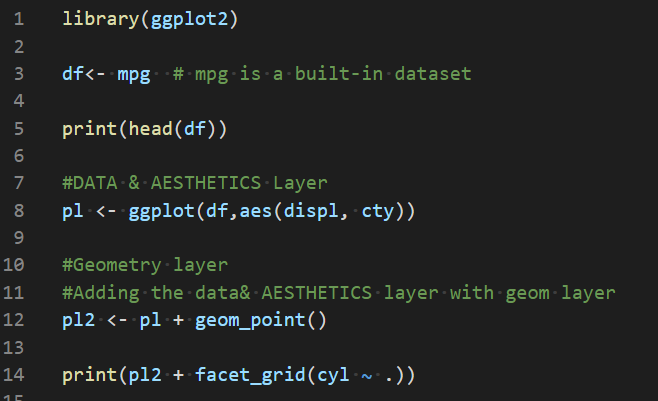
-Save & Run

-o/p 

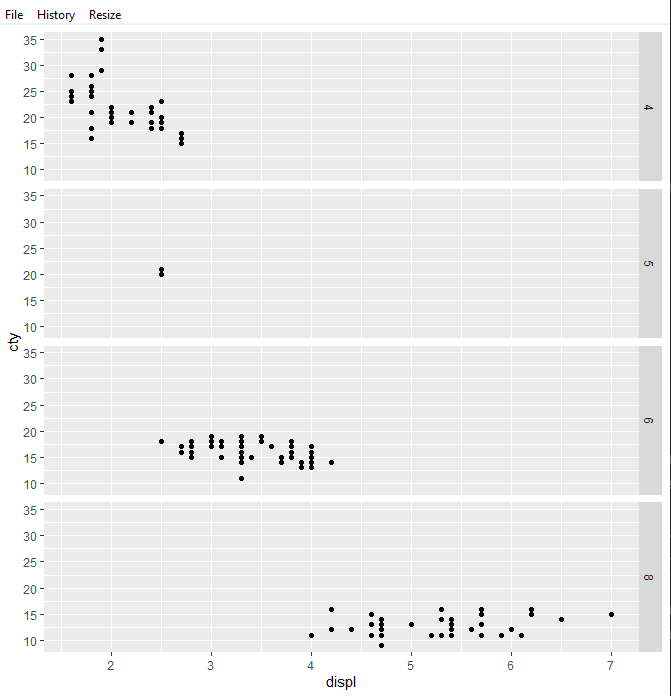
-Explanation: The graph is showing a relation between cty column and displ column when the cyl values 4,5,6 and 8.

We can make the same above relation in another way

-Type the following to update the previous code.



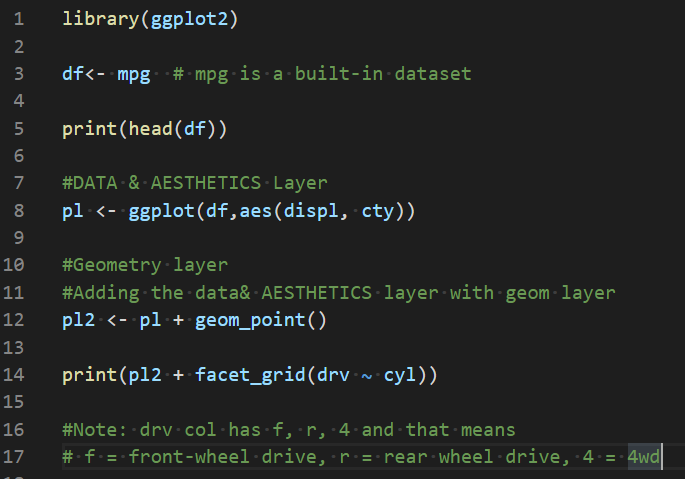
-Run & Save

o/p

**You should be in:**

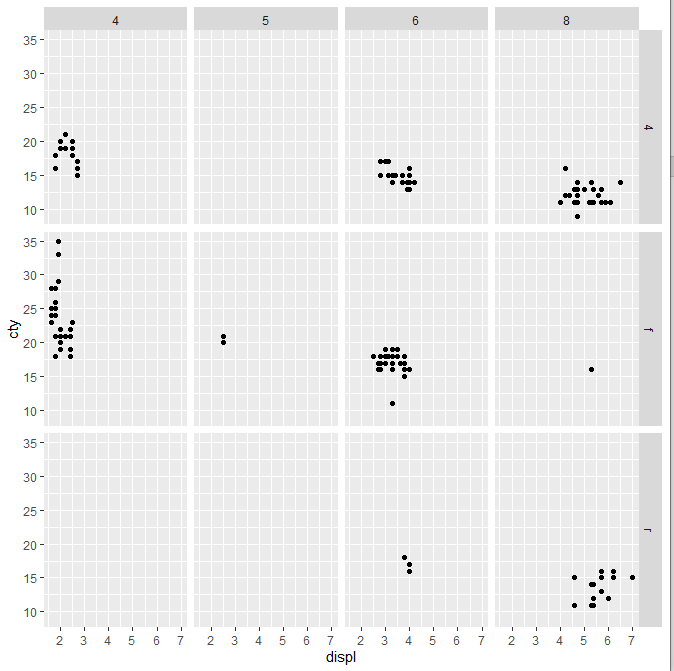
* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3 project folder, create new file DV-FaceGrid2.R**

**Add the following to DV-FaceGrid2.R**



-Save & Run

O/p



**What do the empty cells in plot with facet\_grid(drv ~ cyl) mean? How do they relate to this plot?**

Answer: The facets are empty when there is no data for the according combination e.g. rear wheel drive (r) with 4 or 5 cylinders is not listed. The 7-cylinder factes are missed entirely.

**Themes**

Themes and theme elements control the non-data components of the plot. The use, of predefined themes and custom themes in ggplot.

There are a lot of built-in themes in ggplot and you can use them in two ways, by stating before your plot to set the theme:

theme\_set(theme\_bw())

or by adding them to your plot directly:

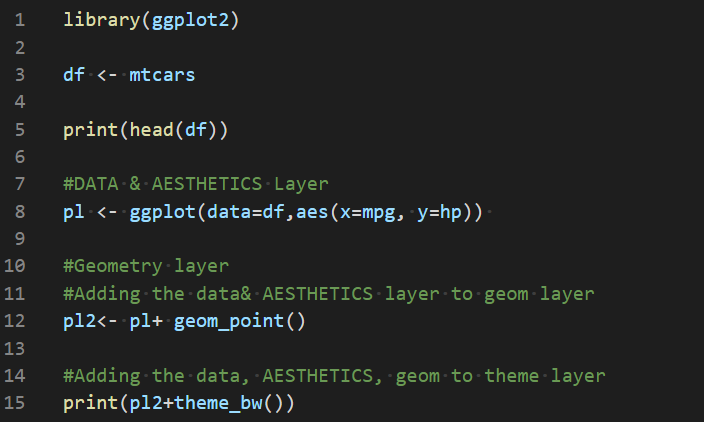
my\_plot + theme\_bw()

There is also a great library called ggthemes which adds even more built-in themes for ggplot. You can also customize your own themes.

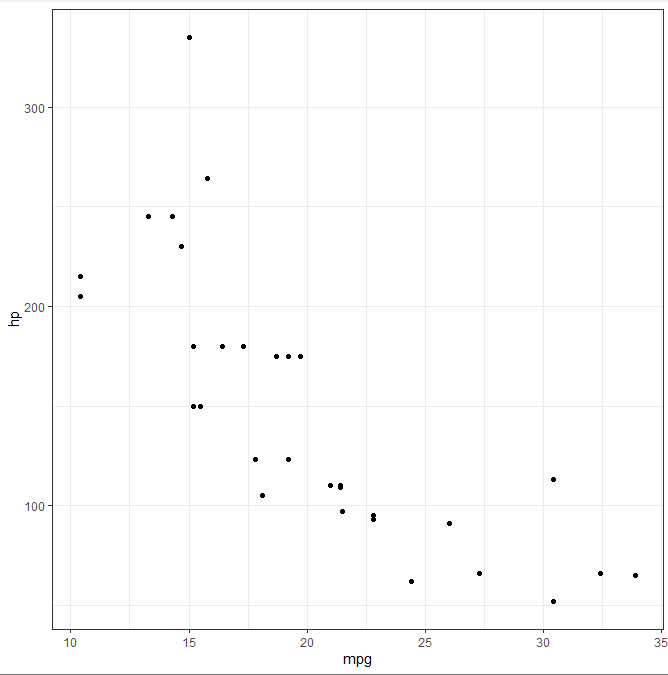
**You should be in:**

* **onsite student:** CS251\_ Fall \_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3 project folder, create new file DV-Themes.R**

**Add the following to DV-Themes.R**



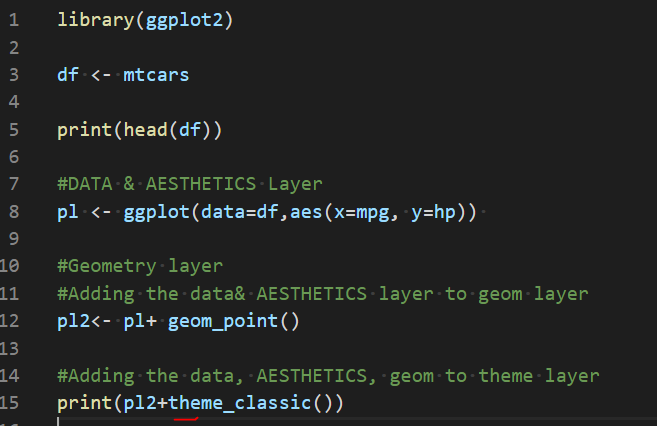
-Save & Run

-O/P 

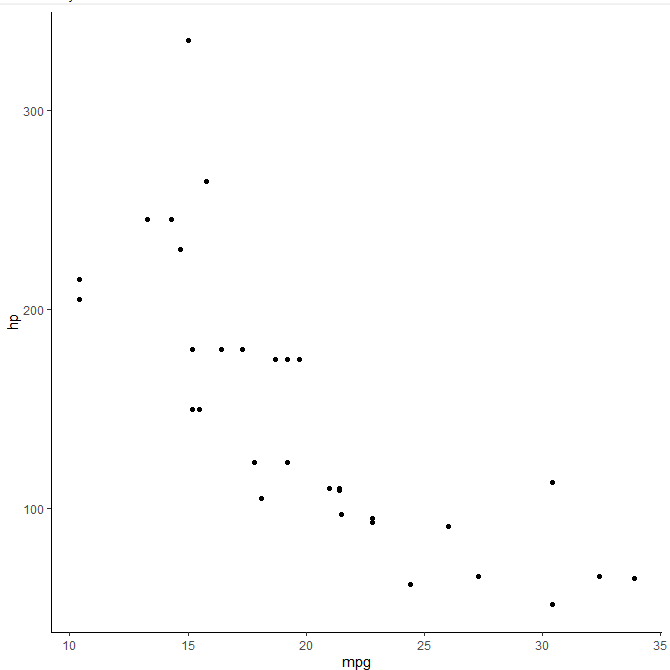
-You will notice that the graph is black and white because we used theme\_bw()

- Let’s try another theme

-Update the previous code with the following

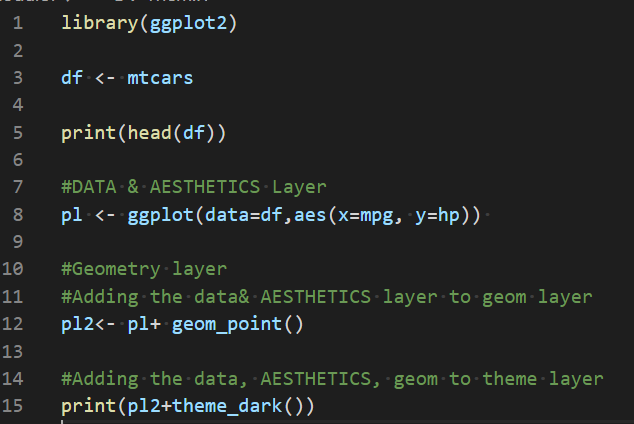


-Save & Run

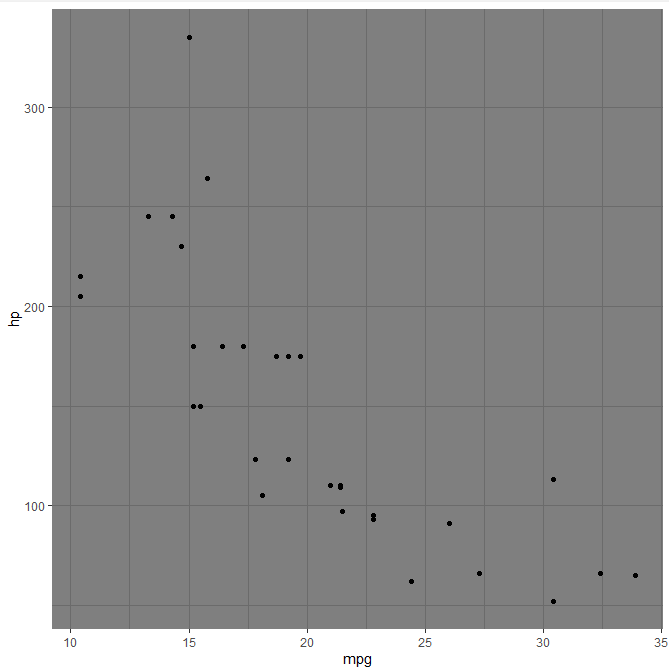


-Let’s try the dark theme

-Update the previous code with the following

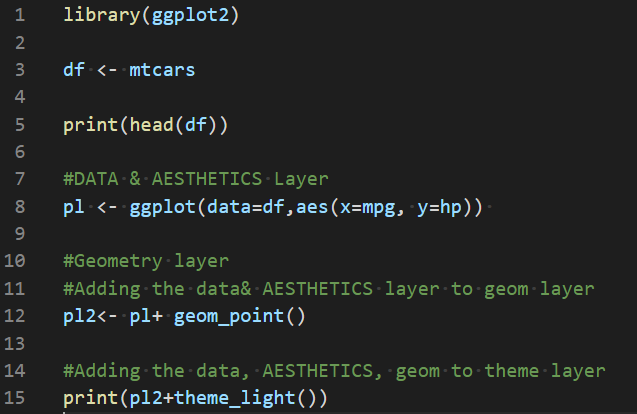


-Save & Run

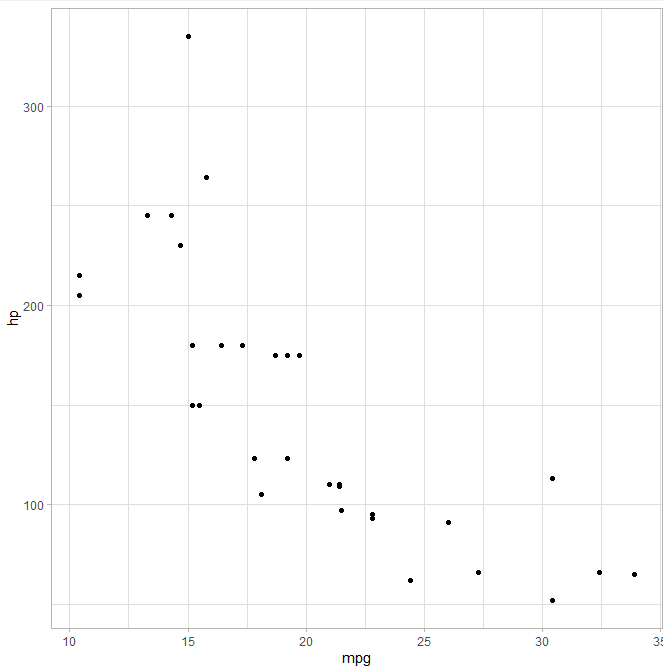
o/p: 

-Let’s try the theme\_light

-Update the previous code with the following

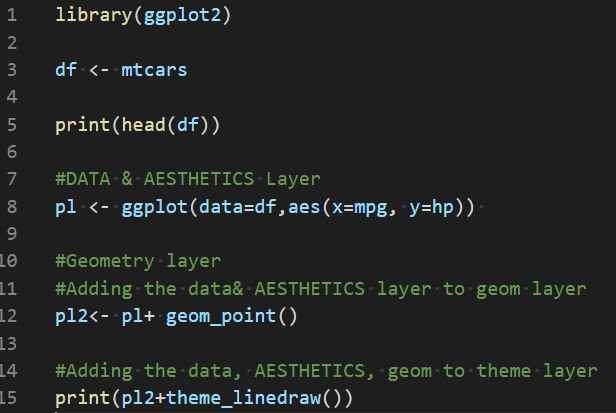


-Save & Run

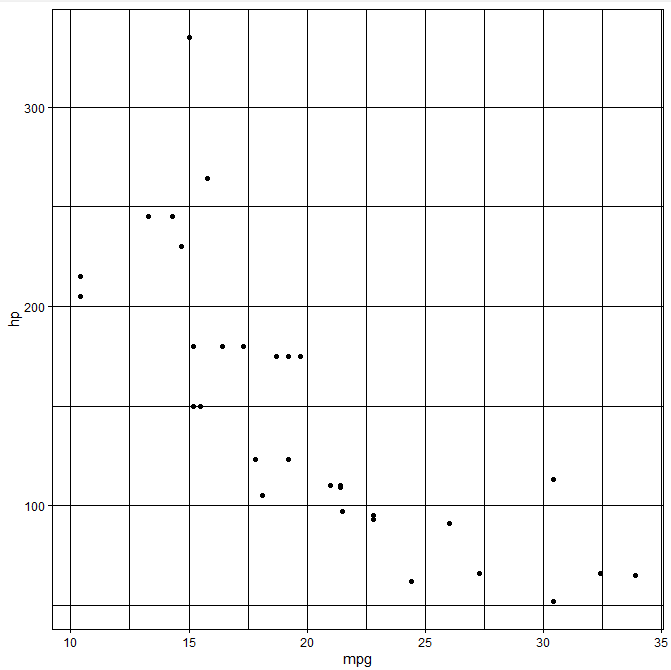


-Let’s try one more theme

-Update the previous code

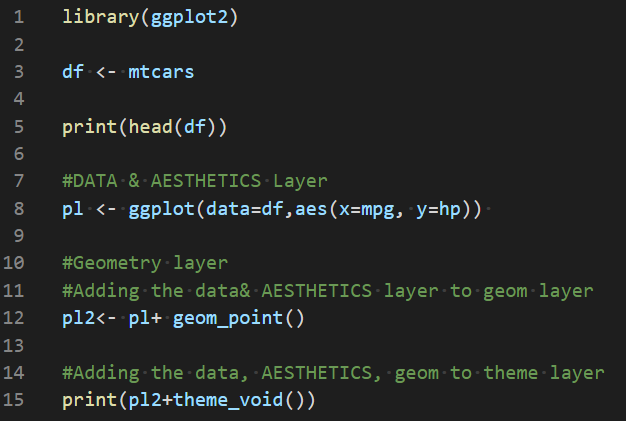


-Save & Run

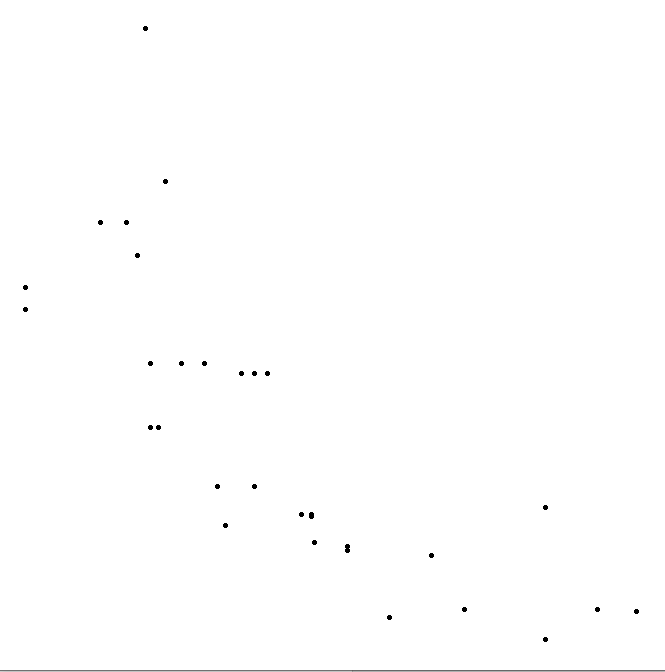
o/p 

-Let’s try the void theme

-Update the previous code



-Save & Run

-o/p 

**ggthemes**

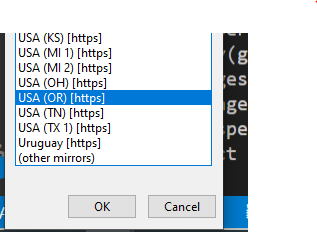
ggthemes provides some extra themes, geoms, and scales for 'ggplot2'. Provides 'ggplot2' themes and scales

To install ggthemes, type in the R console the following

>install.packages(‘ggthemes’)



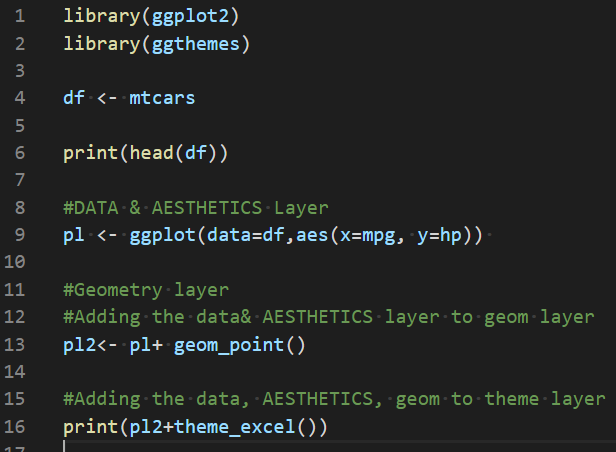
And choose the usa cran



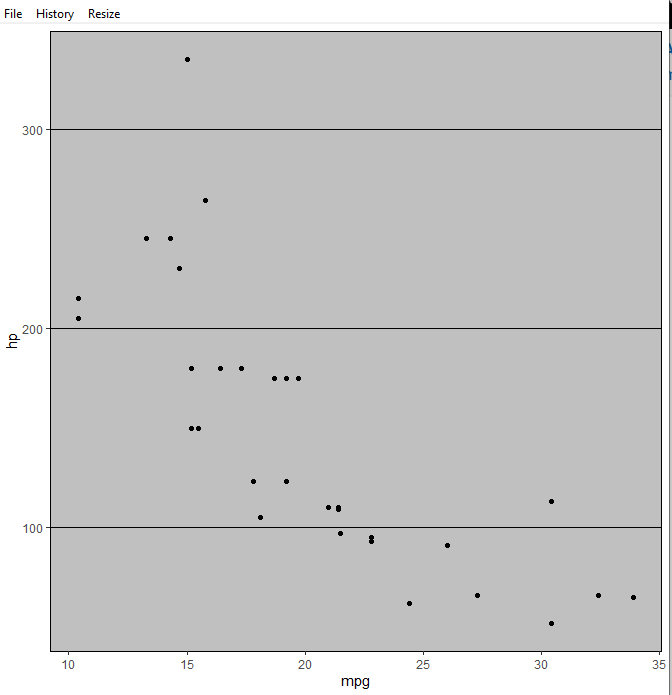
**You should be in:**

* **onsite student:** CS251\_Fall\_2020/**IN**/FirstnameLastname
* **online student:** CS251\_ Fall \_2020/**ON**/FirstnameLastname
* **In Module3 project folder, create new file DV-Themes2.R**

**Add the following to DV-Themes2.R**

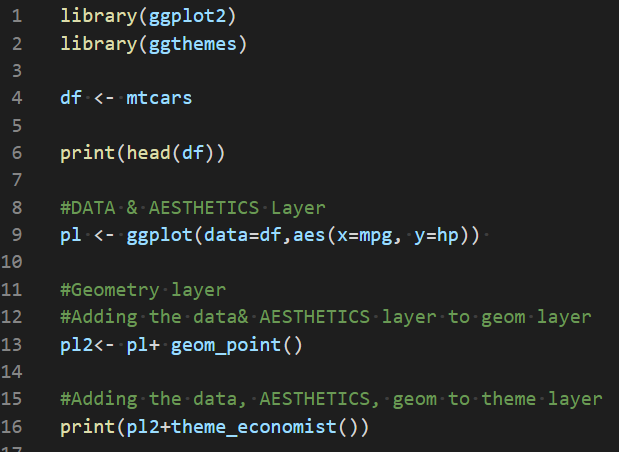


-Save & Run

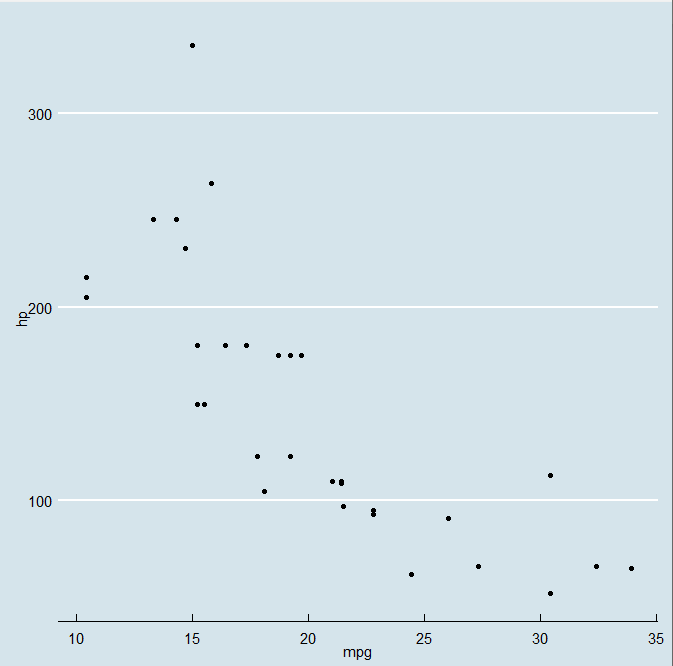
o/p 

**-Let’s try another theme**

-Update the previous code with the following



-Save & Run

o/p 

**Push your work to GitHub**

**Make sure you are in**

Onsite students: CS251\_ Fall \_2020/**IN**/FirstnameLastname

Online students: CS251\_ Fall \_2020/**ON**/FirstnameLastname

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hit the **control + ~** key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 3”

>>> git push origin YOUR\_BRANCH\_NAME

Note: you should change the YOUR\_BRANCH\_NAME to your own branch name. It should be firstname-lastname (e.g. maria-gracia).