

# **Terna Engineering College**

## **Department of Artificial Intelligence and Data Science**

**Program: Sem VI**

**Course: Data Analytics and Visualization Lab**

### **Experiment No.08**

#### **PART A**

**(PART A: TO BE REFERRED BY STUDENTS)**

**A.1 Aim:** To study different visualization experiments in R using various libraries.

#### **A.2 Theory:**

##### **Data Visualization:**

Data visualization is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

##### **Data Visualization packages in R:**

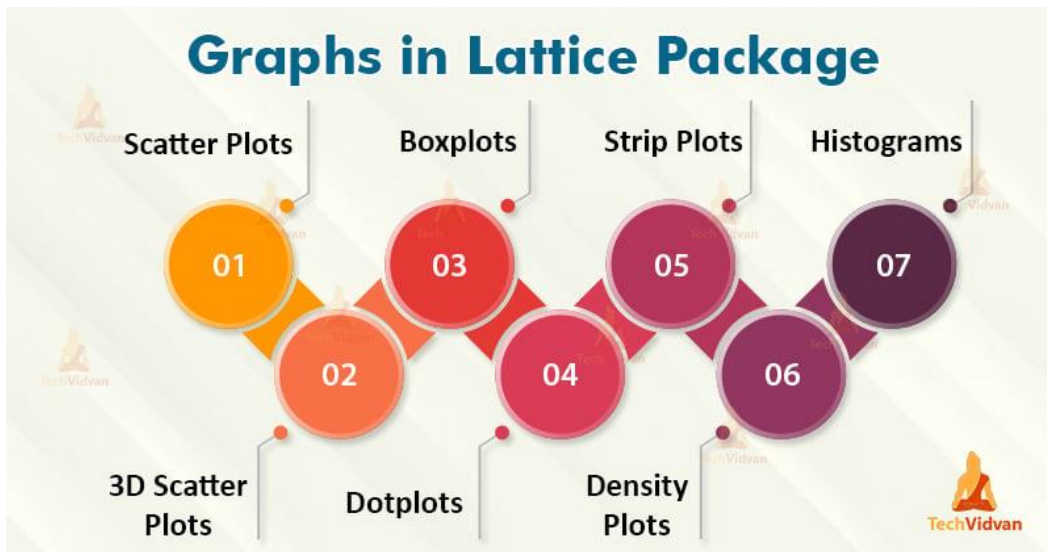
1. **ggplot2:**  
ggplot2 is an R data visualization library that is based on The Grammar of Graphics. ggplot2 can create data visualizations such as bar charts, pie charts, histograms, scatterplots, error charts, etc. using high-level API. It also allows you to add different types of data visualization components or layers in a single visualization. Once ggplot2 has been told which variables to map to which aesthetics in the plot, it does the rest of the work so that the user can focus on interpreting the visualizations and take less time to create them. But this also means that it is not possible to create highly customized graphics in ggplot2. But there are a lot of resources in the RStudio community and Stack Overflow that can provide help in ggplot2 when needed. Just like dplyr, if you want to

install ggplot2, you can install the tidyverse or you can just install ggplot2 using `install.packages("ggplot2")`

## 2. Lattice:

Lattice is a powerful and elegant data visualization package for R programming, with an emphasis on multivariate data. The lattice package is a graphics and data visualization package inspired by the trellis graphics package. The main focus of the package is multivariate data. It has a wide variety of functions that enable it to create basic plots of the base R package as well as enhance on them. Since Lattice is a high-level data visualization library, it can handle many of the typical graphics without needing many customizations. In case you want to extend the capabilities of Lattice, they can download the LatticeExtra package which is an extended version.

We can plot the following types of graphs using the lattice package:



## 3. Plotly:

Plotly is a free open-source graphing library that can be used to form data visualizations. Plotly is an R package that is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual HTML files. Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries. In addition to all this, Plotly can be used offline with no internet connection.

You can install Plotly from CRAN using `install.packages('plotly')` or install the latest development version from GitHub using `devtools::install_github("ropensci/plotly")`.

4. Leaflet:

The Leaflet package is an R interface to the JavaScript Leaflet library that is extremely popular. The Leaflet is very useful in creating interactive but lightweight maps that are seen on various websites such as the Washington Post, the New York Times, etc. There are many useful features in this package such as interactive panning and zooming in the charts, the option to combine Polygons, Lines, Popups, etc. to create charts, embed maps in knitr, create maps in Mercator projections that are non-spherical, and so on. The Leaflet package can be used at the R console after installing it from CRAN using the command `install.packages("leaflet")`.

## PART B

### (PART B: TO BE COMPLETED BY STUDENTS)

*(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)*

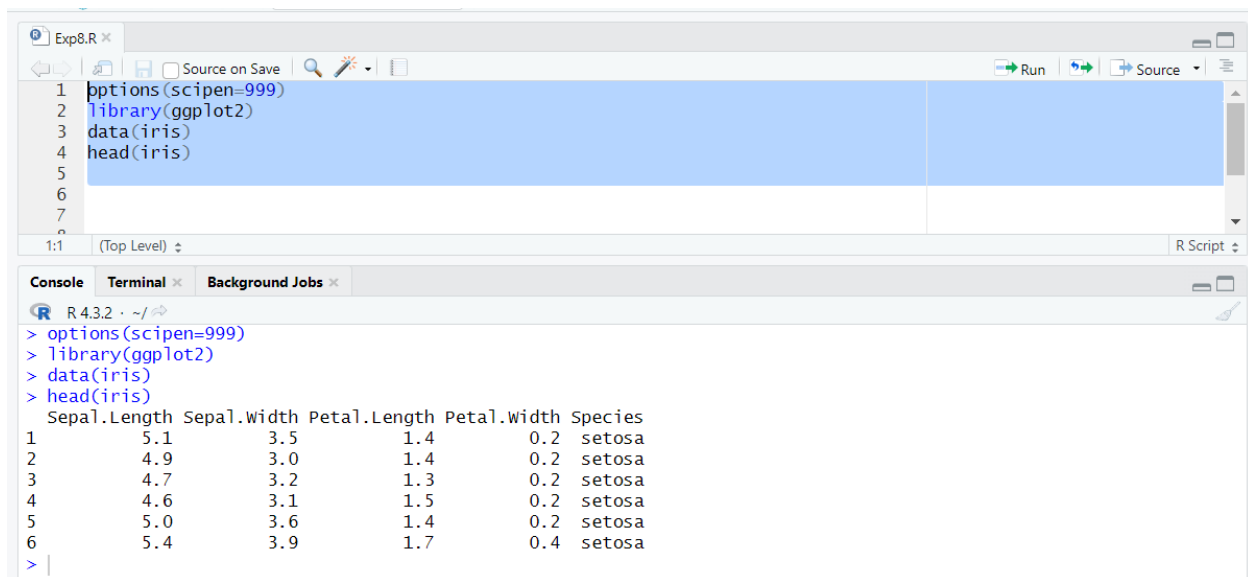
Roll. No. A12	Name: Sufiyan Khan
Class: TE – AI & DS	Batch: A1
Date of Experiment:	Date of Submission: 24-03-24
Grade:	

## B.1 Input and Output:

### ggplot2:

1. Import the ggplot library and load the dataset.

```
options(scipen=999)
library(ggplot2)
data(iris)
head(iris)
```



The screenshot displays the R Studio environment. The script editor at the top contains the following code:

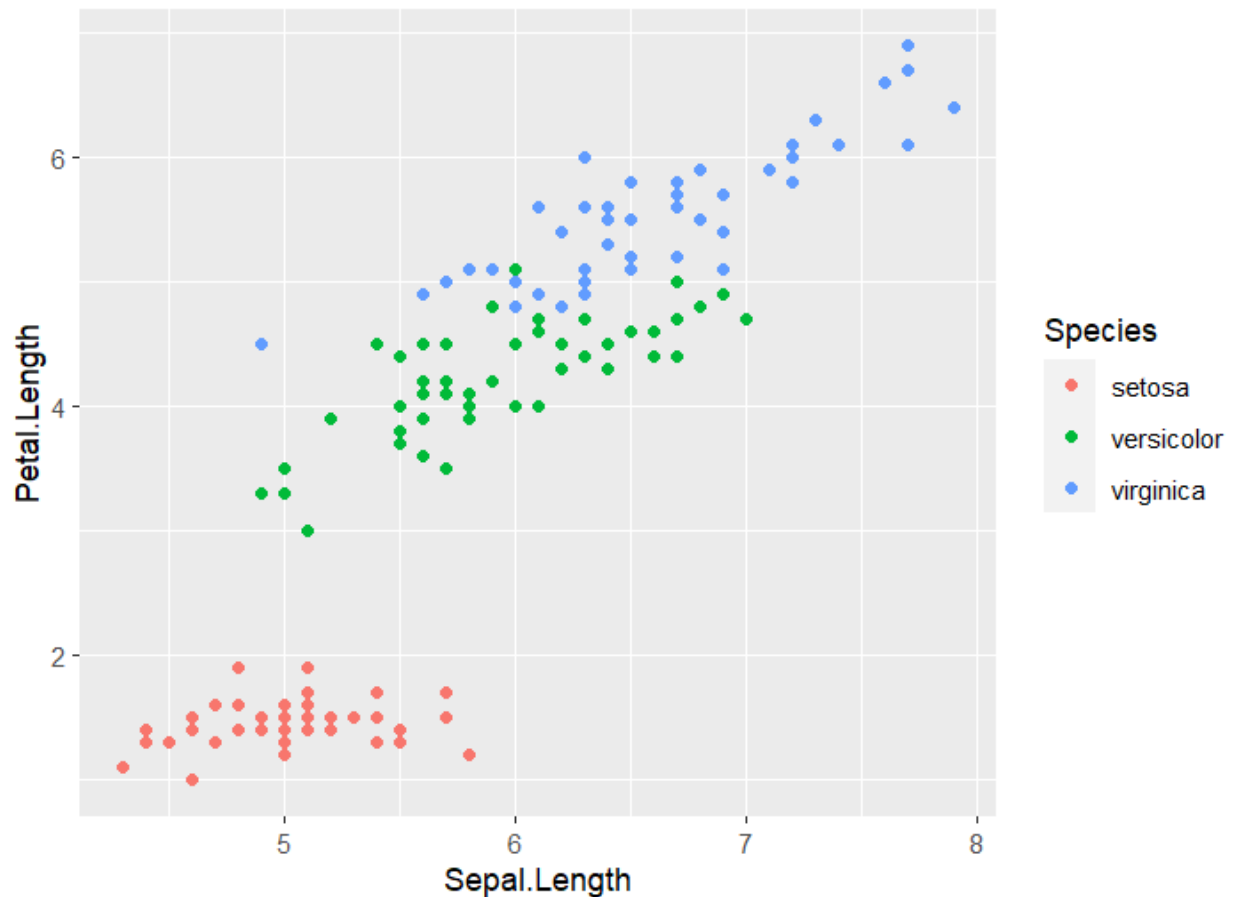
```
1 options(scipen=999)
2 library(ggplot2)
3 data(iris)
4 head(iris)
```

The console at the bottom shows the output of these commands:

```
> options(scipen=999)
> library(ggplot2)
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4          0.2  setosa
2          4.9         3.0          1.4          0.2  setosa
3          4.7         3.2          1.3          0.2  setosa
4          4.6         3.1          1.5          0.2  setosa
5          5.0         3.6          1.4          0.2  setosa
6          5.4         3.9          1.7          0.4  setosa
```

2. To plot a Scatterplot, use the `geom_point()` method. Set X and Y values and provide color = species for differentiating among points from different classes.

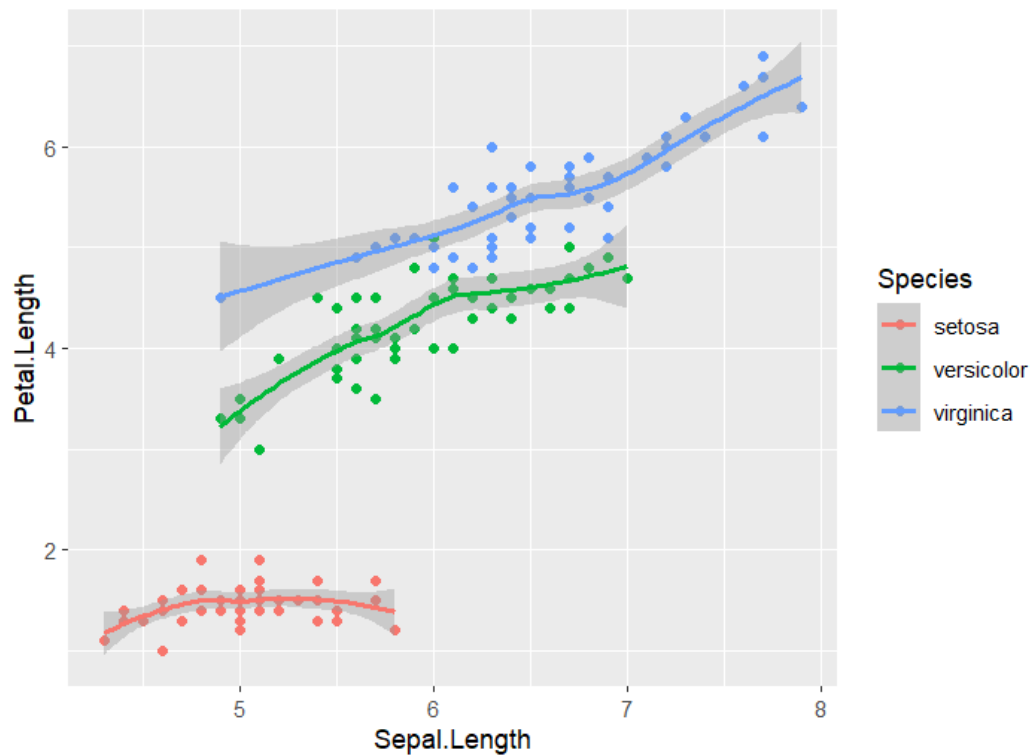
```
ggplot(iris, aes(x=Sepal.Length, y=Petal.Length,
col=Species))+geom_point()
```



3. To visualize linear patterns in data, we use the `geom_smooth()` method with linear model.

```
ggplot(iris, aes(x=Sepal.Length, y=Petal.Length,
col=Species))+geom_point()+geom_smooth()
```

```
> ggplot(iris, aes(x=Sepal.Length, y=Petal.Length, col=Species))+geom_point()
> ggplot(iris, aes(x=Sepal.Length, y=Petal.Length, col=Species))+geom_point()+geom_smooth()
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
> |
```



#### 4. Import mtcars dataset from the library and view its features.

```

Exp8.R
7 ggplot(iris, aes(x=Sepal.Length, y=Petal.Length, col=Species))+geom_point()+geom_smooth()
8
9 data(mtcars)
10 library(tidyverse)
11 glimpse(mtcars)
12
9:1 (Top Level) R Script

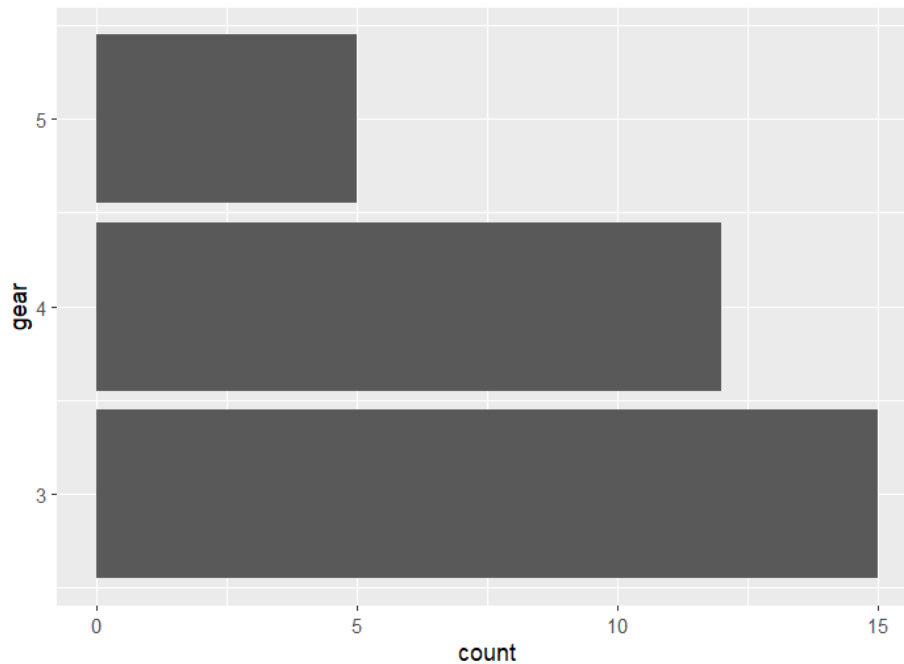
Console Terminal Background Jobs
R 4.3.2 ~ /

> library(tidyverse)
— Attaching core tidyverse packages — tidyverse 2.0.0 —
✓ dplyr 1.1.4 ✓ readr 2.1.5
✓ forcats 1.0.0 ✓ stringr 1.5.1
✓ lubridate 1.9.3 ✓ tibble 3.2.1
✓ purrr 1.0.2 ✓ tidyr 1.3.1
— Conflicts — tidyverse_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag() masks stats::lag()
i Use the conflicted package to force all conflicts to become errors
Warning messages:
1: package 'tidyverse' was built under R version 4.3.3
2: package 'tidyr' was built under R version 4.3.3
3: package 'purrr' was built under R version 4.3.3
4: package 'stringr' was built under R version 4.3.3
5: package 'forcats' was built under R version 4.3.3
6: package 'lubridate' was built under R version 4.3.3
> glimpse(mtcars)
Rows: 32
Columns: 11
$ mpg <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, 16.4, 17.3, 15.2, 10.4, 10.4,...
$ cyl <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 4, 4, 4, 8, 8, 8, 8, 4, 4, 4, 4, 8, 6, 8, 4
$ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 167.6, 167.6, 275.8, 275.8, 275.8,...
$ hp <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180, 205, 215, 230, 66, 52, 65, 9,...
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92, 3.07, 3.07, 3.07, 2.93, 3.00,...
$ wt <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.440, 3.440, 4.070, 3.730, 3.78,...
$ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18.30, 18.90, 17.40, 17.60, 18.0,...
$ vs <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1
$ am <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 4, 5, 5, 5, 5, 5, 4
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2, 2, 4, 2, 1, 2, 2, 4, 6, 8, 2
>

```

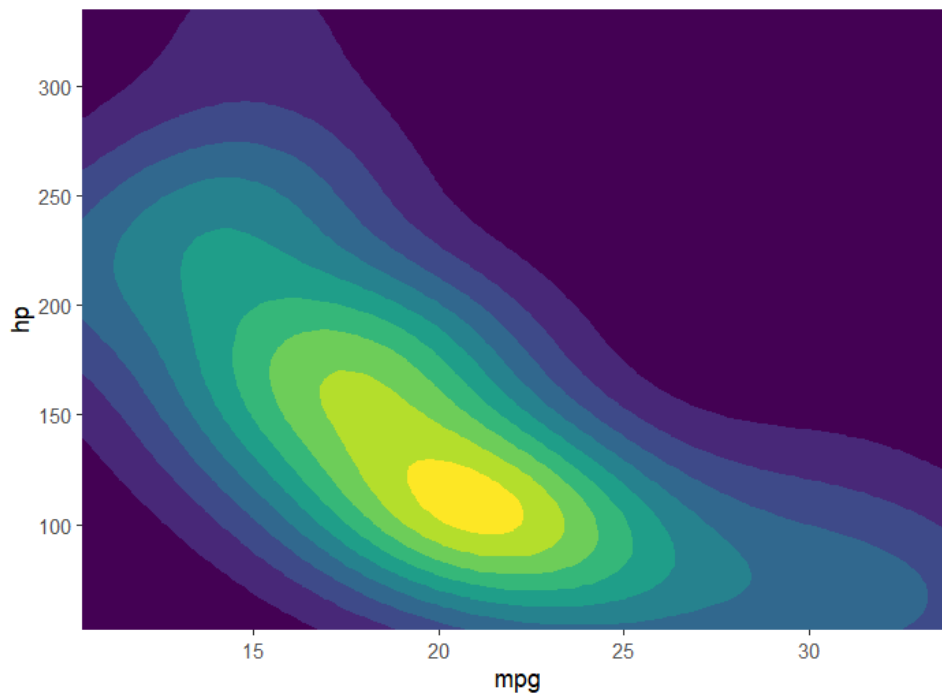
5. To make a barplot, use the `geom_bar()` method. The use of `coord_flip` method rotates the graph to a horizontal view.

```
ggplot(mtcars, aes(x = gear)) + geom_bar() + coord_flip()
```



6. To make a density plot, the `geom_density_2d` method creates a shaded map plot of the data.

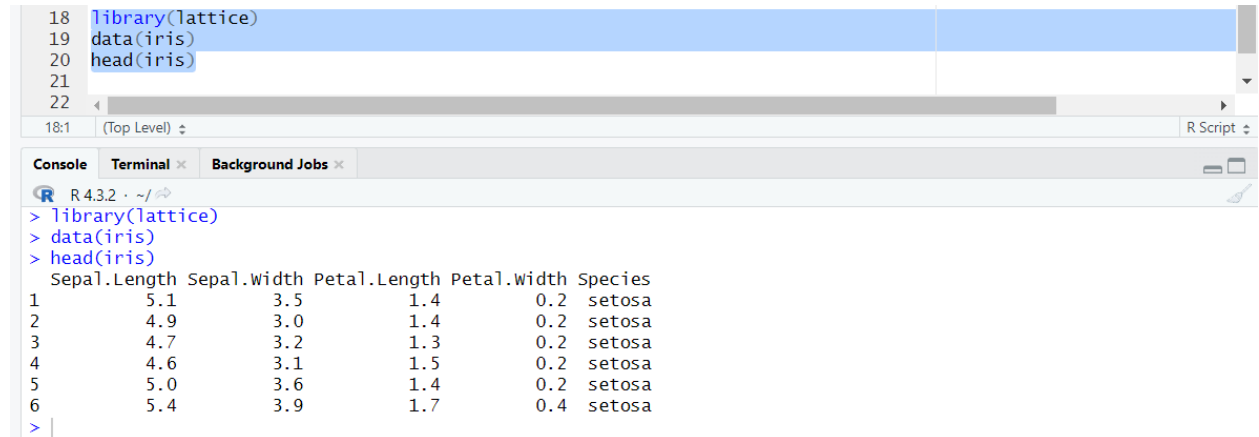
```
ggplot(mtcars, aes(mpg, hp)) +  
geom_density_2d_filled(show.legend = FALSE) +  
coord_cartesian(expand = FALSE) + labs(x = "mpg")
```



## lattice:

1. Firstly we import the lattice package and iris dataset.

```
library(lattice)
data(iris)
head(iris)
```



```
18 library(lattice)
19 data(iris)
20 head(iris)
21
22
```

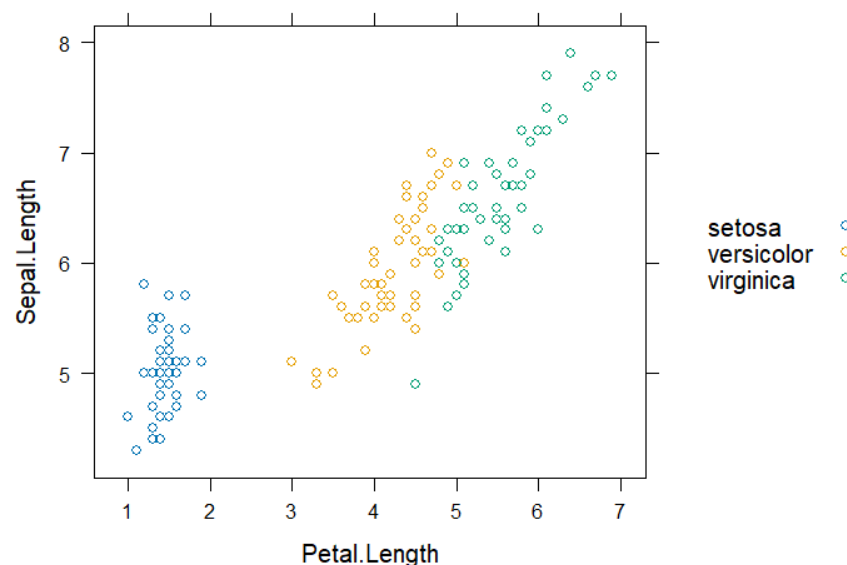
18:1 (Top Level) R Script

Console Terminal Background Jobs

```
R 4.3.2 ~ /
> library(lattice)
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4          0.2  setosa
2          4.9         3.0          1.4          0.2  setosa
3          4.7         3.2          1.3          0.2  setosa
4          4.6         3.1          1.5          0.2  setosa
5          5.0         3.6          1.4          0.2  setosa
6          5.4         3.9          1.7          0.4  setosa
> |
```

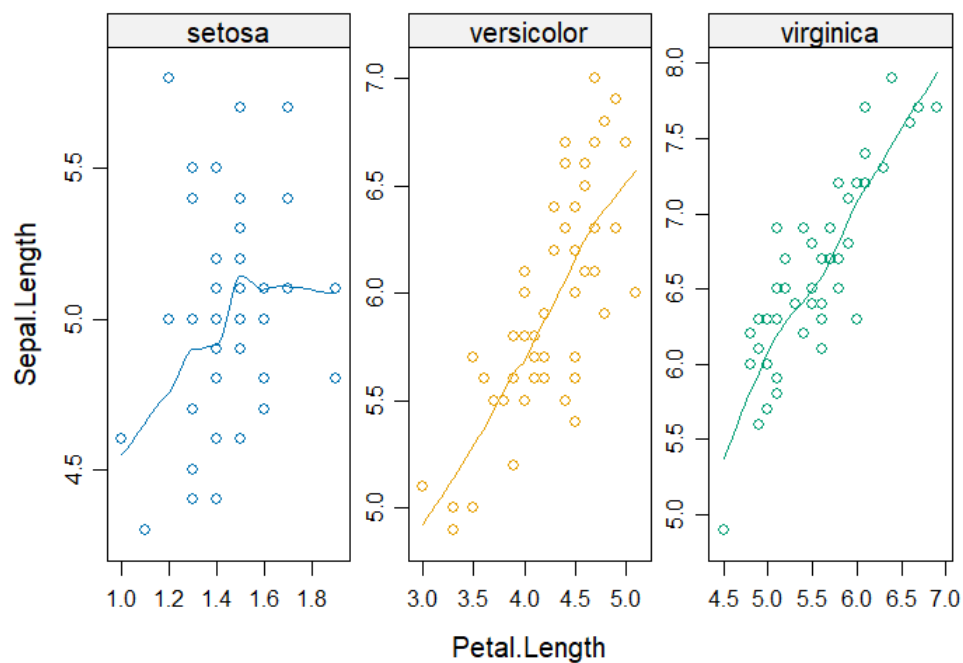
2. To plot a scatter plot, we use the `xyplot()` function. We can add various attributes like labels, type, etc. as well as create multiple graphs in panels based on groups. 3D scatterplots can be created using the `cloud()` function.

```
xyplot(Sepal.Length ~ Petal.Length, data = iris,
       group = Species, auto.key = TRUE)
```

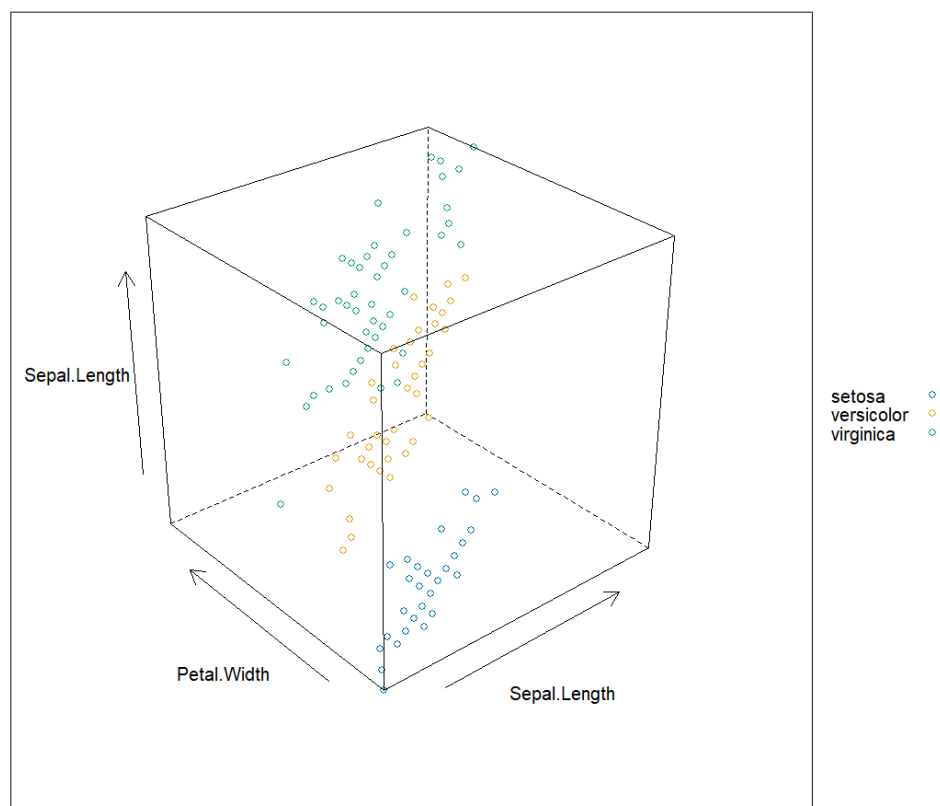




```
xyplot(Sepal.Length ~ Petal.Length | Species, group =
Species,data = iris,type = c("p", "smooth"), scales = "free")
```

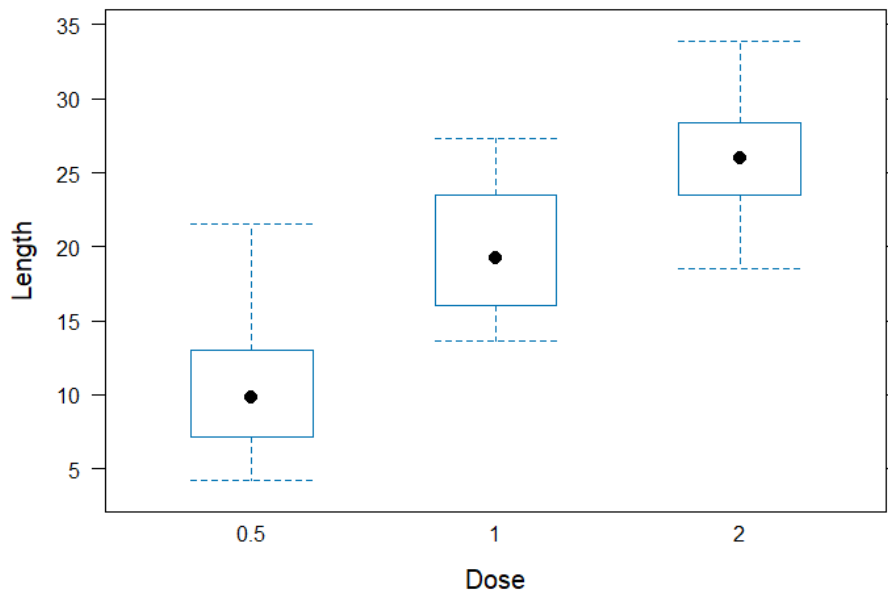


```
cloud(Sepal.Length ~ Sepal.Length*Petal.Width, data = iris,
group = Species, auto.key = TRUE)
```



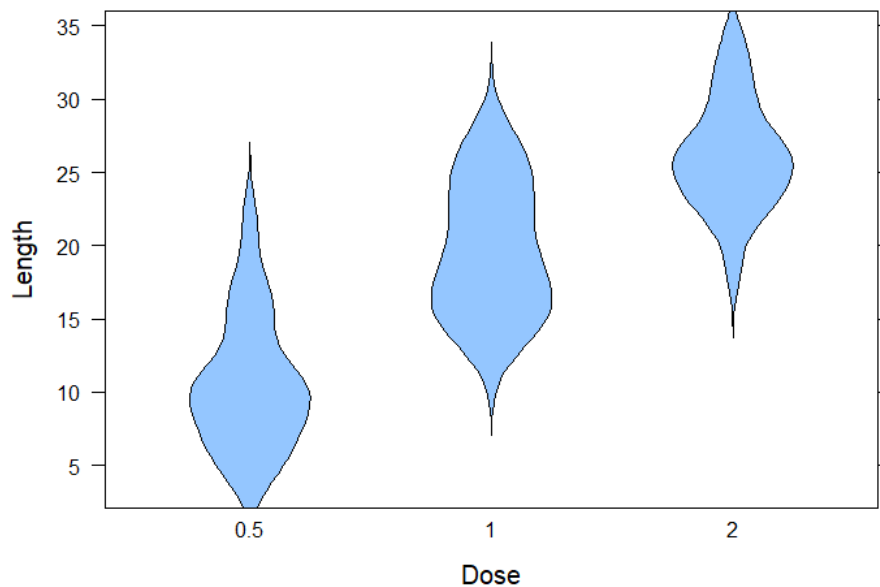
3. To plot a boxplot, the lattice package provides a `bwplot()` function. We will be using the `ToothGrowth` data set for this box plot.

```
ToothGrowth$dose <- as.factor(ToothGrowth$dose)
bwplot(len ~ dose, data = ToothGrowth, xlab = "Dose", ylab =
"Length")
```



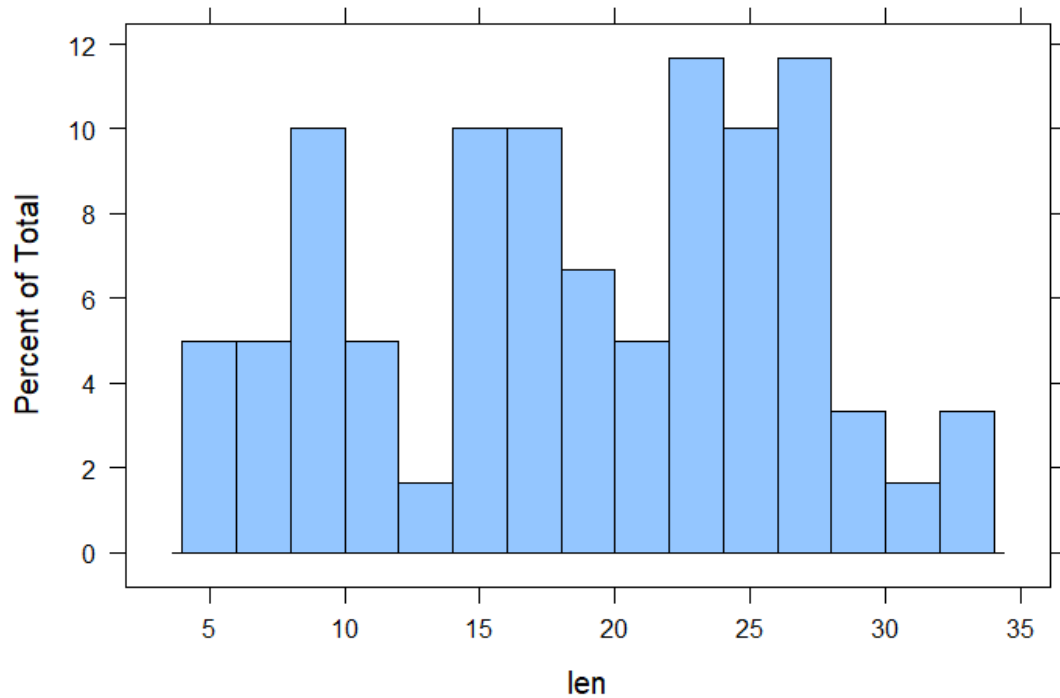
Using the `panel` argument of `bwplot()` function, we can make a violin plot as well.

```
bwplot(len ~ dose, data = ToothGrowth, xlab = "Dose", ylab =
"Length", panel = panel.violin)
```



4. To create a histogram using the lattice package, we can use the `histogram()` function.

```
histogram(~ len, data = ToothGrowth, breaks = 20)
```



## B.2 Conclusion:

Thus we have successfully studied and explored various libraries and packages in R for data visualization. We have learnt how to implement various plots and graphs using libraries like ggplot and lattice.