**DATA SCIENCE**

**LATTICE AND GGPLOT2 PACKAGE**

With ever increasing volume of data, it is impossible to tell stories without visualizations. Data visualization is an art of how to turn numbers into useful information.

R has two high-level graphics system. And some basic plots.

* LATTICE : based on Trellis Graphics.
* GGPLOT2 : inspired by “Grammar of Graphics “.

LATTICE PACKAGE :- It’s a powerful and elegant high level data visualization system. That is being inspired by Terillis graphics. Although, it is being designed with an emphasis on multivariate data that allows easy conditioning to produce “small multiple dots”.

The data set used is Iris. It includes three iris species with 50 sample each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are :

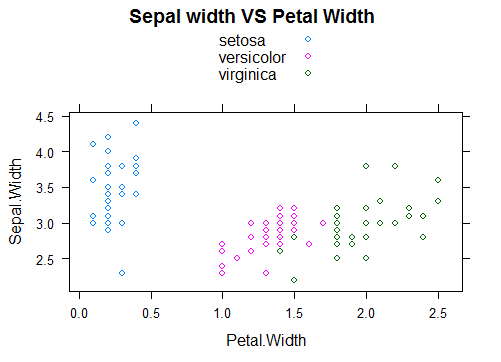
* Id
* Sepal.Length(Cm)
* Sepal.Width(Cm)
* Petal.length(Cm)
* Petal.Width(Cm)
* Species

Functions in lattice :

1. **Xyplot() : scatter plot** .To produce bivariate scatter plots or time series plots .
   * + - The scatter plot is a useful way to visualize the relationship between two variables.
       - Used to plot relationship between 2 numeric variables.
       - They are used to make initial diagnoses before any statistical analysis. The simplified format as follow :

**Syntax :**xyplot(x, data,auto.key = FALSE, aspect = "fill",groups = NULL, xlab, xlim, ylab, ylim)

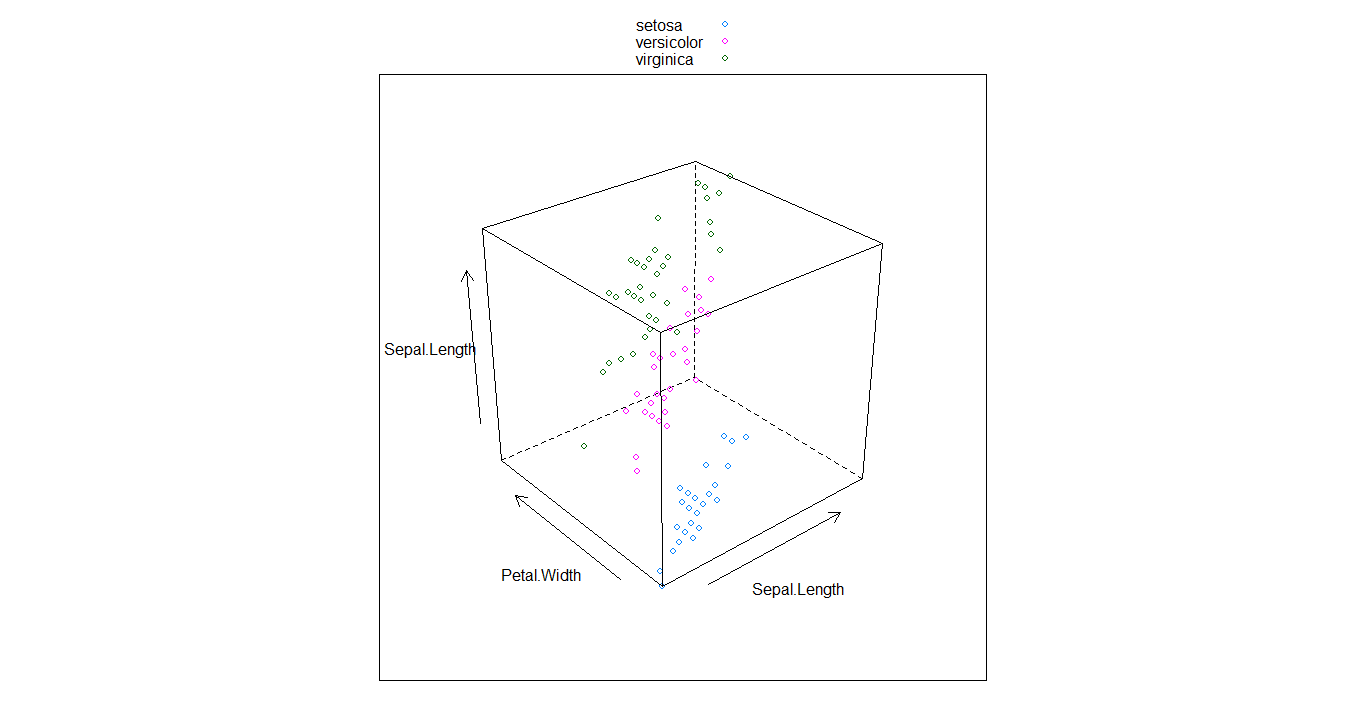
**code :**

xyplot(Sepal.Length ~ Petal.Length, groups = Species, data =train, auto.key = TRUE, main = " Sepal Length VS Petal Length")

1. **Cloud()** : **3D scatter plot**: generic function to draw 3d scatter plot .
   * + Plots continuous variables

**syntax :**cloud(x, data,xlab,ylab,zlab, xlim = if (is.factor(x)) levels(x) else range(x, finite = TRUE), ylim = if (is.factor(y)) levels(y) else range(y, finite = TRUE), zlim = if (is.factor(z)) levels(z) else range(z, finite = TRUE),

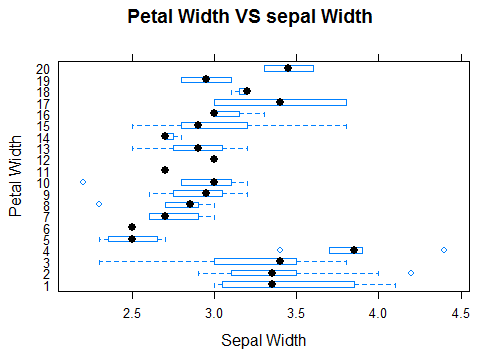
**CODE** :*cloud(Sepal.Length ~ Sepal.Length \* Petal.Width, groups = Species , data = train , auto.key = TRUE)*



1. **Bwplot() :**the function bwplot() makes box and whisker plots for **numerical variables.** 
   * + It can be used to summarize the data.

**Syntax :**bwplot(x,data, auto.key = FALSE, aspect = "fill", groups = NULL, xlab, xlim, ylab, ylim,horizontal = NULL)

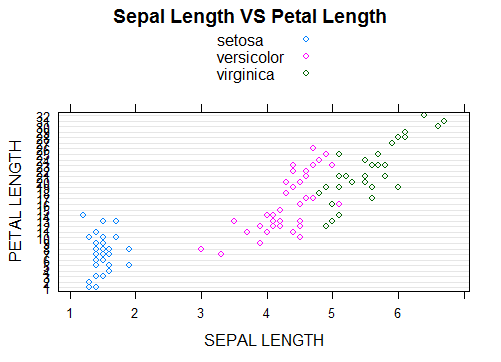
**CODE :***bwplot(Petal.Width ~ Sepal.Width , data = train , xlab = "Sepal Width", ylab = "Petal Width" , main = "Petal Width VS sepal Width")*



1. **Dotplot() : dot plot.**To produce dot plots .
   * + works on numeric data. It is an alternative to bar charts where the bars are replaced by dots.

**Syntax :***dotplot(x, data)*

**Code** .dotplot(Sepal.Width ~ Petal.Width, data = train,group = Species, auto.key = TRUE, xlab = "SEPAL WIDTH", ylab = "PETAL WIDTH", main = "Sepal Width VS Petal Width")

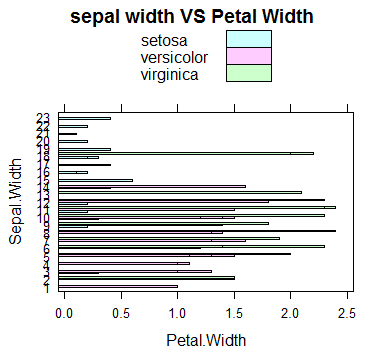


1. **Barchart():** Bar charts are used to compare things between different groups or track changes over time
   * + Best to use when changes are larger.
     + Used for ordinal and nominal datesets.

**syntax :**barchart(x, data,box.ratio = 2)

**CODE :**

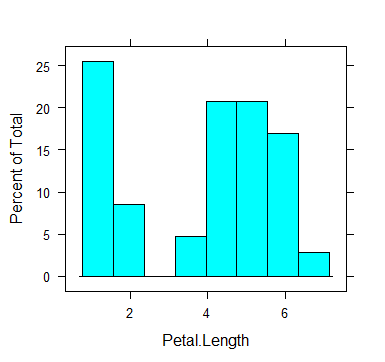
barchart( ~ Petal.Length , data = train , auto.key = TRUE,group = Species , main = " Petal Length")



1. **Histogram() :** Histograms are used to plot the frequency of score occurrences in a continuous data set that has been divided into classes, called bins.

**Syntax** :histogram(x,data, auto.key = FALSE,groups,xlab, xlim, ylab, ylim,)

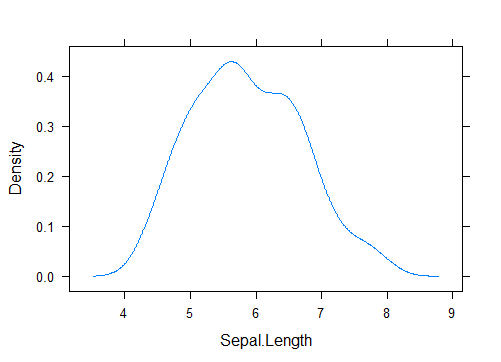
**Code** :histogram(~Petal.Width, data = train)



1. **Densityplot() :.** To produce density plot.
   * + - This plot visualizes the distribution of data over a continuous interval of time period.
       - This chart is a variation of a histogram that uses a kernel smoothing to plot values, allowing for smoother distributions by smoothing out the noise.

**Syntax :**densityplot(x data,xlab, xlim, ylab, ylim, bw, adjust, kernel, window, width)

**CODE:***densityplot(~Sepal.Length, group = Species, data = train, plot.points = FALSE, auto.key = TRUE*



**GGPLOT2 :**the ggplot2 package is one of the most widely used visualizations with little code using **grammar of graphics**  . The 1**grammar of graphics** is a general scheme for data visualizations which breaks up graphs into semantic components such scales and layers.

A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

The popularity of ggplot2 has increased tremendously in recent years since it makes it possible to create graphs that contain both univariate and multivariate data in a very simple manner.

|  |  |
| --- | --- |
| Components | Description |
| Data | The dataset being plotted |
| Aesthetic | The scales onto which we plot our data |
| Geometry | The visual elements used or data. |
| Facet | Groups by which we divide the data |

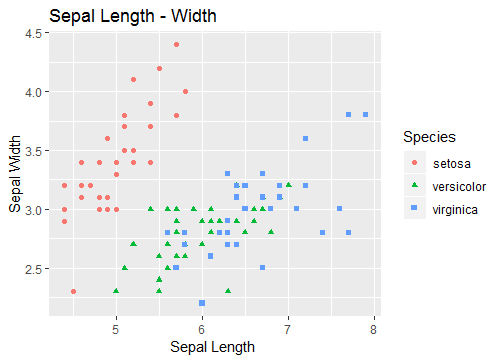
Functions in ggplot2 :

We will use ggplot() function .

ggplot(data = train, aes(x= Petal.Length, y = Sepal.Length, color = Sepal.Length))

1. **Scatter plot :**The point geom is used to create scatterplots.
   * + The scatterplot is most useful for displaying the relationship between two continuous variables.
     + It can be used to compare one continuous and one categorical variable,

**Syntax :**geom\_point(mapping = NULL, data = NULL, stat = "identity",position = "identity", ..., na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

CODE :*geom\_point() + xlab("Sepal Length ") + ylab("petal Length") + ggtitle("petal Length - Sepal Length")+Scale\_color\_gradientn(colors=rainbow(4))*

From this we can conclude that specie having low sepal length and higher sepal width is SETOSA

Specie having medium sepal length and sepal width is VERSICOLOR.

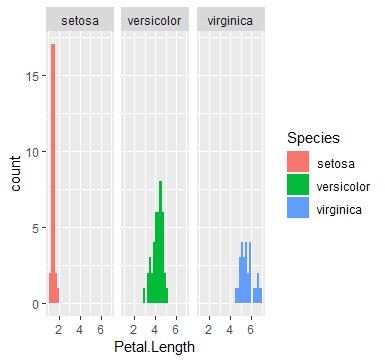
Specie having higher sepal Length and medium sepal width is VERGINICA.

1. **Histogram :**  to plot two continuous variable .it breaks the data into bins and shows frequency distribution of these bins. We need to show only one aesthetic.

**Syntax :**histogram(x,data, auto.key = FALSE,groups,xlab, xlim, ylab, ylim,)

**CODE :**

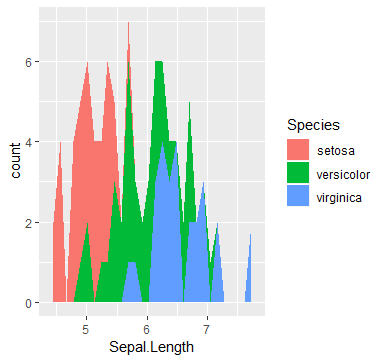
geom\_histogram() + facet\_wrap(~Species)



1. **Area chart :** used to show continuity across a variable or a dataset. It is very much same as a line chart and is commonly used for time series plot. Alternatively , it is used to plot continuous variables and analyze the underlying trends.

**Syntax :**geom\_area(mapping = NULL, data = NULL, stat = "identity", position = "stack", na.rm = FALSE, show.legend = NA, inherit.aes = TRUE, ...)

**Code :**geom\_area( stat = "bin")

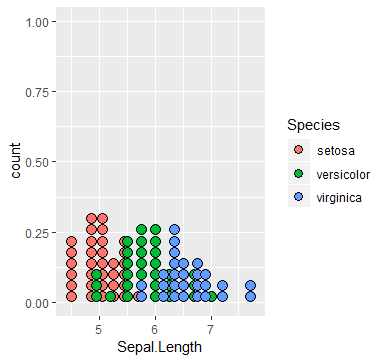


1. **Dot plot :**theR ggplot2 dot plot, or dot chart consists of a data point drawn on a specified scale. Done using geom\_dotplot() function .

**Syntax :**geom\_dotplot(mapping = NULL, data = NULL, position = "identity", ...,

binwidth = NULL

**CODE :**geom\_dotplot(binwidth = 0.2)

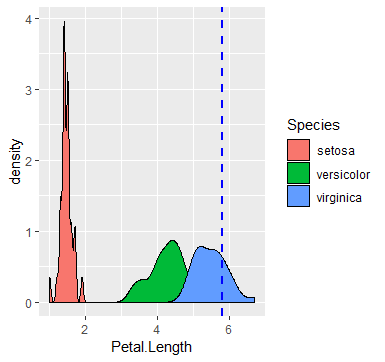


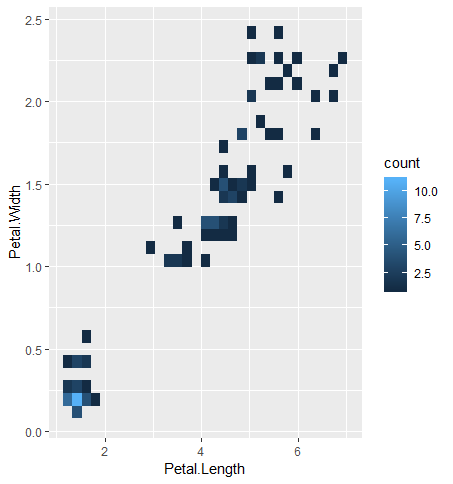
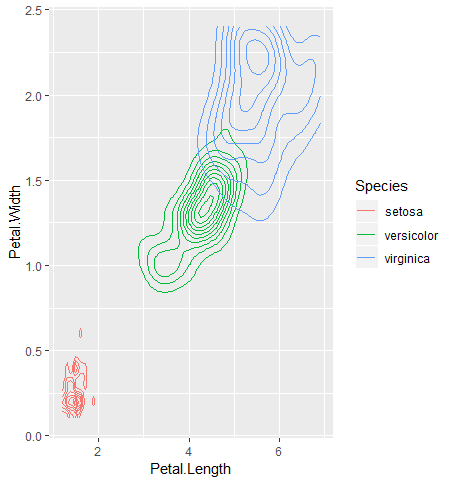
1. **Density plot :**used to show the density of a variable using geom\_density() function. we can also show the mean using geom\_vline() function

**Syntax :**geom\_density(mapping = NULL, data = NULL, stat = "density")

**CODE: 1.** geom\_density() + geom\_vline(aes(xintercept = mean(Sepal.Length)) , color = "blue", linetype = "dashed" , size = 1 )

**2.**  geom\_density2d()

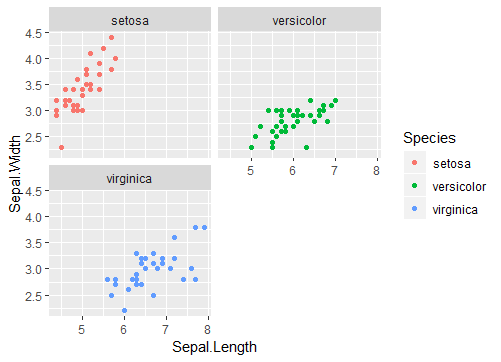
**3.** geom\_bin2d()  




1. **Facet wrap :**facet\_wrap wraps a 1d sequence of panels into 2d. this is better use of screen space then facet\_grid because most displays are roughly. We use this using facet\_wrap () function.

**Syntax :**facet\_wrap(facets, nrow = NULL, ncol = NULL, scales = "fixed")

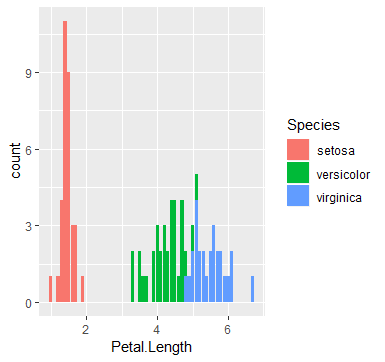
**CODE**: geom\_point() + facet\_wrap(~Species , nrow = 2 , ncol = 2)



7. Bar chart :geom\_bar() makes the height of the bar proportional to the number of cases in each group (or if the weight aesthetic is supplied, the sum of the weights).

Syntax :geom\_bar(mapping = NULL, data = NULL, stat = "count",

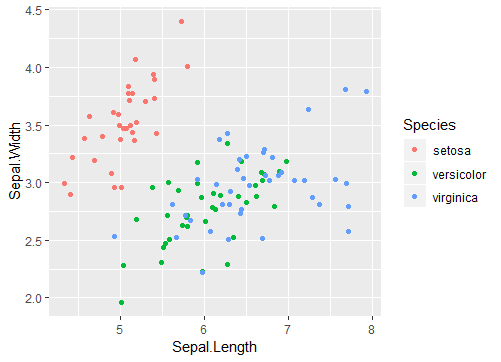
Code :ggplot(data = train , aes(Petal.Length ,fill = Species )) + geom\_bar(stat = "bin")



1. **Jitter** :The ggplot2 jitter is useful to handle the over plotting caused by the smaller dataset discreteness using geom\_jitter() function.

**Syntax** :geom\_jitter(mapping = NULL, data = NULL, stat = "identity", position = "jitter", ..., width = NULL, height = NULL, na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

**Code**: geom\_jitter(aes(color = Species))

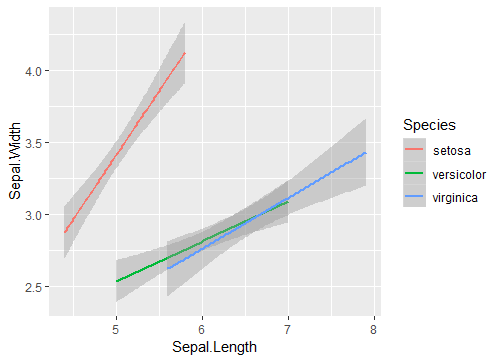


1. **Smoothplot: :**smoothed conditional means aids the eye in seeing pattern in the presence of overplotting.
   * + the general idea is to group data points that are expected to have similar expectations and compute the average.
     + Plotted over numeric data .

Done using geom\_smooth() function .

**Syntax:** geom\_smooth(mapping = NULL, data = NULL, stat = "smooth", position = "identity", ..., method = "auto", formula = y ~ x, se = TRUE, na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

**Code** *:geom\_smooth(method = lm)*



**Basic plots :**

we have some high level graphics unction that starts a new graph . it initializes the graph window ; sets the scale and may draw some adornments, such as a title and labels; and renders the graphic.

Some of them are :

* Plot : generic plotting function
* Boxplot : create a box plot.
* Hist : create a histogram.
* Qqnorm : create a quantile – quantile (Q –Q) plot,
* Curve : graph a function

We do have some low level functions as well which cannot a start a new graph but add something to an existing one.

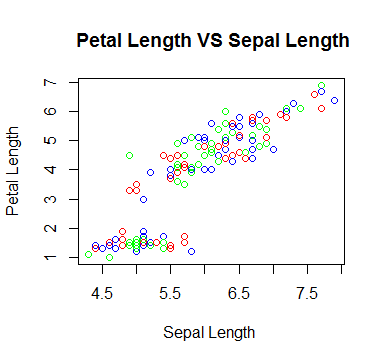
* Points : add point
* Lines : add lines
* Abline: add a straight line
* Segments: add line segment
* Polygon : add a closed polygon
* Text : add text.

1. **Scatter plot** : a scatter plot provide a graphical view of the relationship between two sets of numbers.
   * + Plots relationship between two numeric dataset.

We use function**plot()** function.

**Syntax :**plot(x, y, ...)

**CODE :**plot(x= iris$Sepal.Length, y= iris$Petal.Length, xlab = "Sepal Length ", ylab = "Petal Length", main = " Petal Length VS Sepal Length", col = c("red","green ", "blue"))

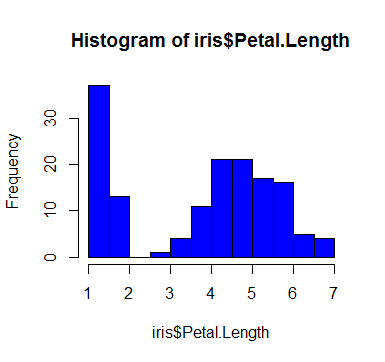


1. **Histogram**: The generic function hist computes a histogram of the given data values. If plot = TRUE, the resulting object of [class](http://127.0.0.1:22583/help/library/graphics/help/class)"histogram" is plotted by [plot.histogram](http://127.0.0.1:22583/help/library/graphics/help/plot.histogram), before it is returned.
   * + to plot two continuous variable .

**Syntax :**hist(x, main = paste("Histogram of" , xname), xlab , = xname, ylab)

**CODE** :

hist(iris$Petal.Length, col = "blue")



1. **Dot plot :**Draw a Cleveland dot plot. It is an alternative to bar chart that works on continuous dataset.

**Syntax :**dotchart(x, labels = NULL, groups = NULL, gdata = NULL,

cex = par("cex"), pt.cex = cex,

pch = 21, gpch = 21, bg = par("bg"),

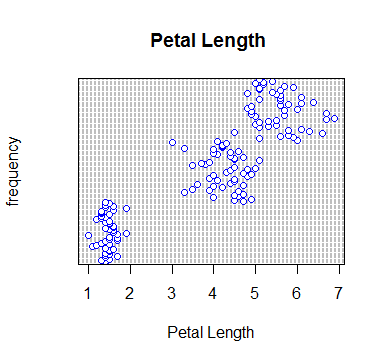
color = par("fg"), gcolor = par("fg"), lcolor = "gray",

xlim = range(x[is.finite(x)]),

main = NULL, xlab = NULL, ylab = NULl

**CODE :**

dotchart(iris$Petal.Length ,ylab = "frequency", xlab = "Petal Length", main= "Petal Length ", color = "blue")



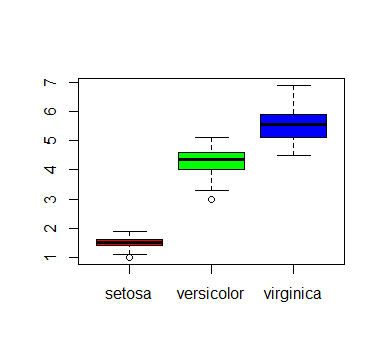
1. **Box plot :** Produce box-and-whisker plot(s) of the given (grouped) values used to show the shape of distribution, its central value and its variability,

**Syntax :**boxplot(formula, data = NULL, ..., subset, na.action = NULL,

drop = FALSE, sep = ".", lex.order = FALSE)

**CODE :**

*boxplot(iris$Petal.Length ~ iris$Species , col = c("red", "green", "blue"))*



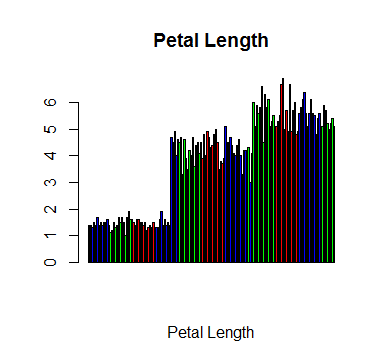
In this the middle lines show the medians.

1. **Bar chart** :Creates a bar plot with vertical or horizontal bars. Plots continuous variables.

**Syntax** :barplot(height, width = 1, space = NULL,names.arg = NULL, legend.text = NULL, beside = FALSE,horiz = FALSE,main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, xpd = TRUE )

**CODE :**

barplot(iris$Petal.Length, col = c("red", "green","blue"), xlab = "Petal Length" , main = "Petal Length")



**Difference between LATTICE and GGPLOT2**

|  |  |
| --- | --- |
| **LATTICE** | **GGPLOT2** |
| * Most useful for conditioning types of plots:   Looking at how y changes with x across levels of z. | * Split the difference between base and lattice |
| * Things like margins/spacing set automatically because entire plot is specified at once. | * Automatically deals with spacing ,text, tiles but also allows you to annotate by adding |
| * Good for putting many plots on the screen. | * Superficial similarity to lattice but generally easier/more intuitive to use. |
| 3D plot is there. | * Default mode makes many choices for us (but we can customize!) |

**Different types plots:**

|  |  |  |
| --- | --- | --- |
| * **Basic Plots** | * **Lattice** | * **Gglot2** |
| * **Scatter plot :** | * **Scatter plot** | * **Scatter plot** |
| * **Histogram** | * **Histogram** | * **Histogram** |
| * **Dotplot** | * **Dot plot** | * **Dot plot** |
| * **bar plot** | * **Bar plot** | * **bar plot** |
| * **box plot** | * **box plot** | * **box plot** |
|  | * **density plot** | * **density plot** |

**Comparisons between plots :**

**BASIC PLOTS :**

* The base plotting paradigm is “ink on paper” whereas the lattice and ggplot2 paradigm are basically writing program that used the grid package to accomplish the low level output to the target graphics package.
* There’s no r object that holds results. The command gets processed immediately and inscribed on the paper of the current device .

**LATTICE :**

* Not being actively maintained, but it seemed fairly mature at the point that active development was stopped and if you find a bug it will probably be fixed.
* gridExtra and latticeExtra packages extend lattice capabilities.

**GGPLOT2 :**

* With ggplot2 you can assign the result of the function to an object name and then further modify it. When it is ready for publication you get the output processed and sent to a device with print.
* Ggplot graphics often get progressively modified by adding layers to a base plot created with qplot or ggplot through the use of +gg() function .
* It does not have any 3D function .