R Graphics

R Line

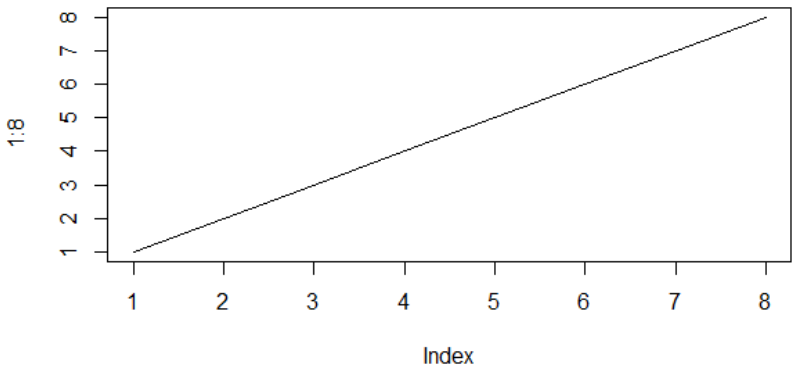
Line Graphs

A line graph has a line that connects all the points in a diagram.

To create a line, use the plot() function and add the type parameter with a value of "l":

Example:

```
> plot(1:8, type="l")
```

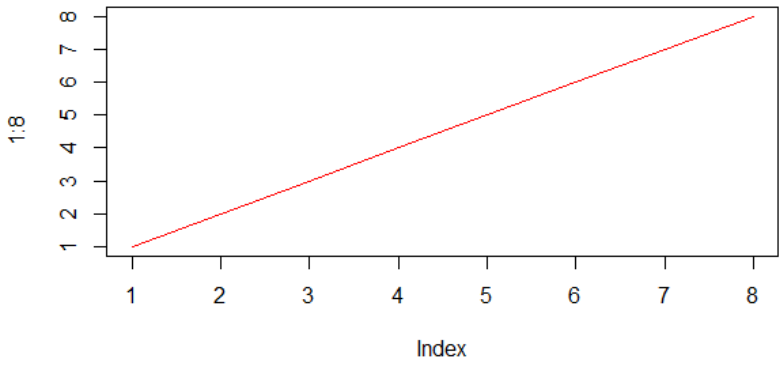


Line Color

The line color is black by default. To change the color, use the col parameter:

Example:

```
> plot(1:8, type="l", col="red")
```

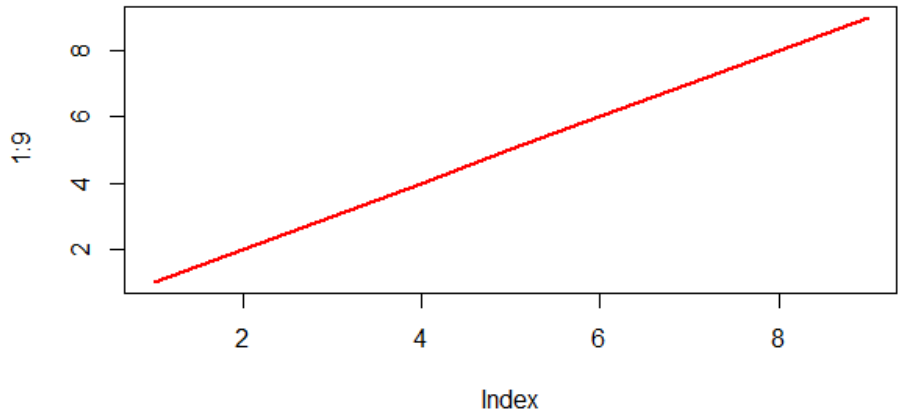


Line Width

To change the width of the line, use the lwd parameter (1 is default, while 0.5 means 50% smaller, and 2 means 100% larger):

Example:

```
plot(1:9, type="l", col="red", lwd=2)
```



Line Styles

The line is solid by default. Use the lty parameter with a value from **0 to 6** to specify the line format.

For example, lty=3 will display a dotted line instead of a solid line:

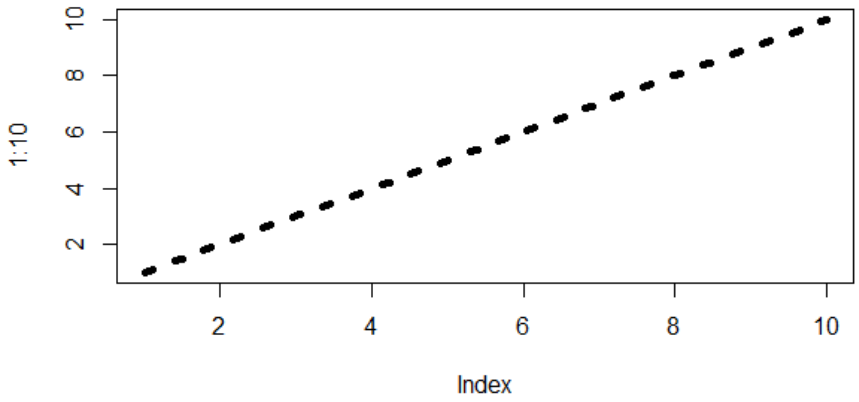
Example:

```
plot(1:10, type="l", lwd=5, lty=3)
```

Available parameter values for lty:

- 0 removes the line
- 1 displays a solid line
- 2 displays a dashed line
- 3 displays a dotted line
- 4 displays a "dot dashed" line
- 5 displays a "long dashed" line
- 6 displays a "two dashed" line

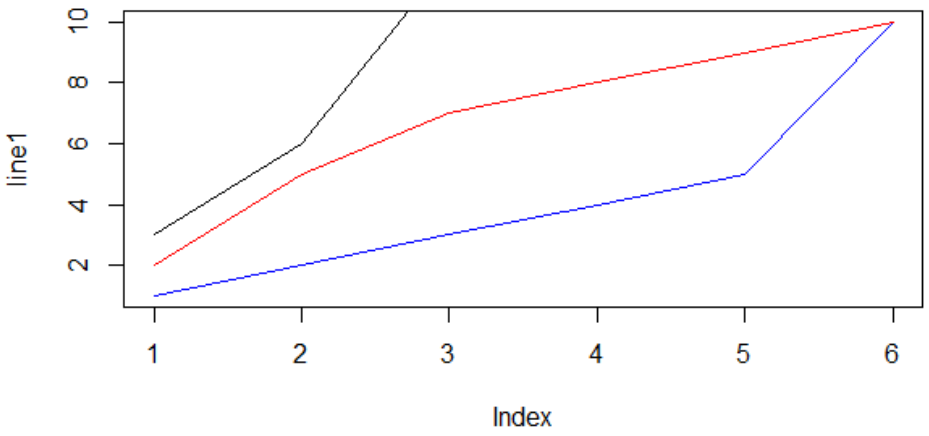
Multiple Lines



To display more than one line in a graph, use the plot() function together wit h the lines() function:

Example:

```
> line1 <- c(1,2,3,4,5,10)
> line2 <- c(2,5,7,8,9,10)
> line3 <- c(3, 6, 12)
>
> plot(line1, type = "l", col = "blue")
> lines(line2, type="l", col = "red")
> lines(line3, type="l", col ="black")
```



R Scatter Plot

Scatter Plots

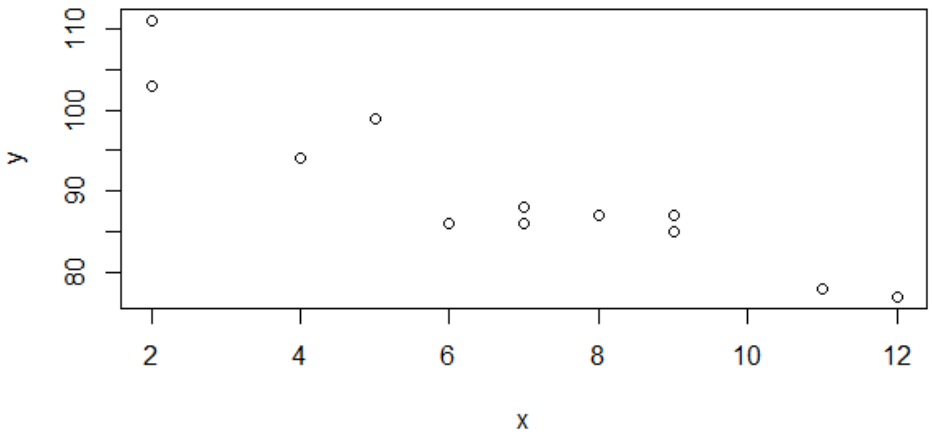
You learned from the Plot part that the plot() function is used to plot numbers against each other.

A "scatter plot" is a type of plot used to display the relationship between two numerical variables, and plots one dot for each observation.

It needs two vectors of same length, one for the x-axis (horizontal) and one for the y-axis (vertical):

Example:

```
> x <- c(5,7,8,7,2,2,9,4,11,12,9,6)
> y <- c(99,86,87,88,111,103,87,94,78,77,85,86)
>
> plot(x, y)
```



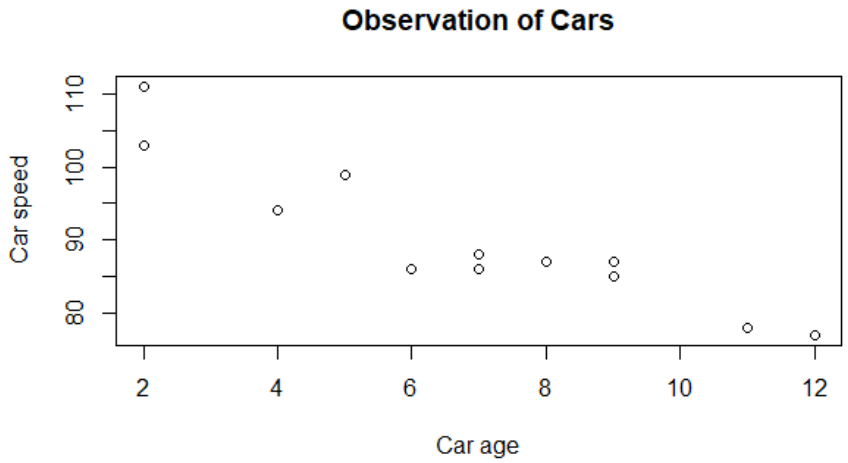
The observation in the example above should show the result of 12 cars passing by.

That might not be clear for someone who sees the graph for the first time, so let's add a header and different labels to describe the scatter plot better:

Example

```
> x <- c(5,7,8,7,2,2,9,4,11,12,9,6)
```

```
> y <- c(99,86,87,88,111,103,87,94,78,77,85,86)
>
> plot(x, y, main="Observation of Cars", xlab="Car age", yl
ab="Car speed")
```



To recap, the observation in the example above is the result of 12 cars passing by.

The **x-axis** shows how old the car is.

The **y-axis** shows the speed of the car when it passes.

Are there any relationships between the observations?

It seems that the newer the car, the faster it drives, but that could be a coincidence, after all we only registered 12 cars.

Compare Plots

In the example above, there seems to be a relationship between the car speed and age, but what if we plot the observations from another day as well? Will the scatter plot tell us something else?

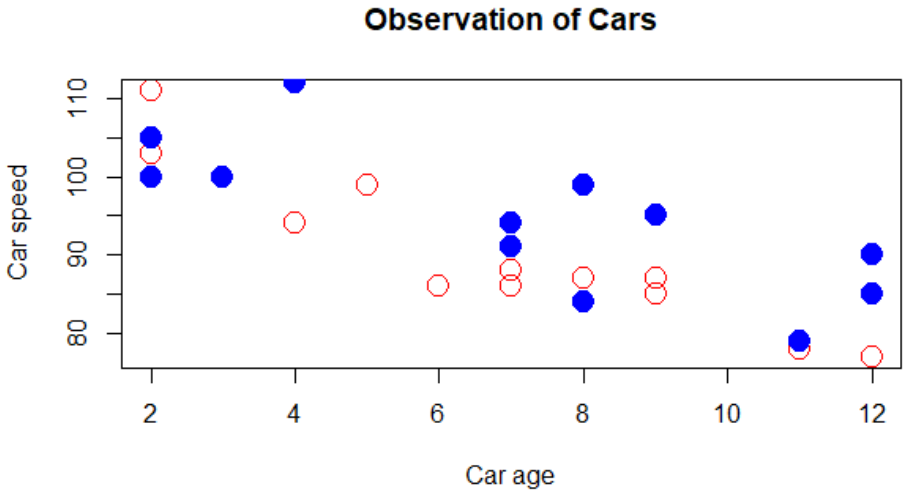
To compare the plot with another plot, use the points() function:

Example

Draw two plots on the same figure:

```
> # day one, the age and speed of 12 cars:
> x1 <- c(5,7,8,7,2,2,9,4,11,12,9,6)
```

```
> y1 <- c(99,86,87,88,111,103,87,94,78,77,85,86)
>
> # day two, the age and speed of 15 cars:
> x2 <- c(2,2,8,1,15,8,12,9,7,3,11,4,7,14,12)
> y2 <- c(100,105,84,105,90,99,90,95,94,100,79,112,91,80,85
)
>
> plot(x1, y1, main="Observation of Cars", xlab="Car age",
ylab="Car speed", col="red", cex=2)
> points(x2, y2, col="blue", cex=2, pch=19)
```



Note: To be able to see the difference of the comparison, you must assign different colors to the plots (by using the col parameter). Red represents the values of day 1, while blue represents day 2. Note that we have also added the cex parameter to increase the size of the dots.

Conclusion of observation: By comparing the two plots, I think it is safe to say that they both gives us the same conclusion: the newer the car, the faster it drives.

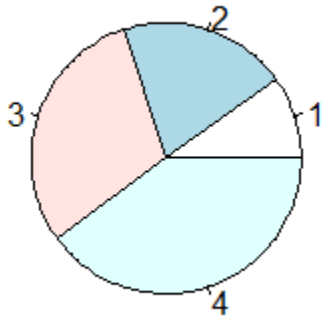
R Pie Charts

Pie Charts

A pie chart is a circular graphical view of data. Use the pie() function to draw pie charts:

Example:

```
> # Create a vector of pies
> x <- c(10,20,30,40)
>
> # Display the pie chart
> pie(x)
```



Example Explanation

As you can see the pie chart draws one pie for each value in the vector (in this case 10, 20, 30, 40).

By default, the plotting of the first pie starts from the x-axis and move **counterclockwise**.

Note: The size of each pie is determined by comparing the value with all the other values, by using this formula:

The value divided by the sum of all values: x/sum(x)

Start Angle

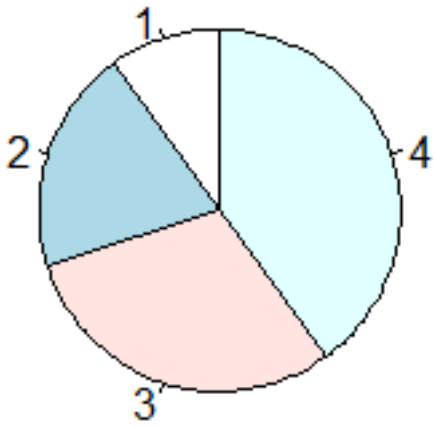
You can change the start angle of the pie chart with the init.angle parameter.

The value of init.angle is defined with angle in degrees, where default angle is 0.

Example

Start the first pie at 90 degrees:

```
> # Create a vector of pies
> x <- c(10,20,30,40)
>
> # Display the pie chart and start the first pie at 90 deg
rees
> pie(x, init.angle = 90)
```

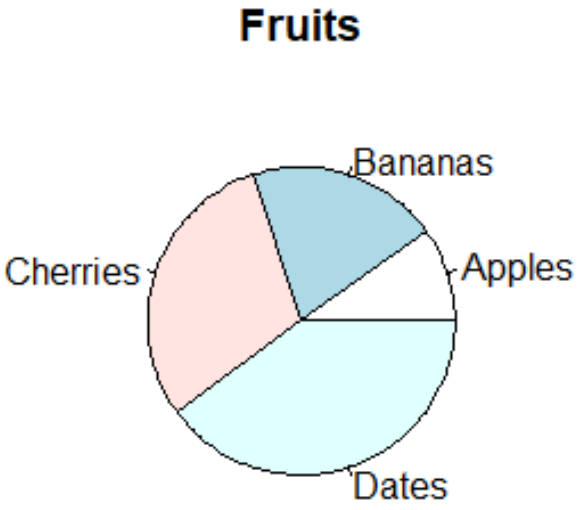


Labels and Header

Use the label parameter to add a label to the pie chart, and use the main parameter to add a header:

Example:

```
> # Create a vector of pies
> x <- c(10,20,30,40)
>
> # Create a vector of labels
> mylabel <- c("Apples", "Bananas", "Cherries", "Dates")
>
> # Display the pie chart with labels
> pie(x, label = mylabel, main = "Fruits")
```

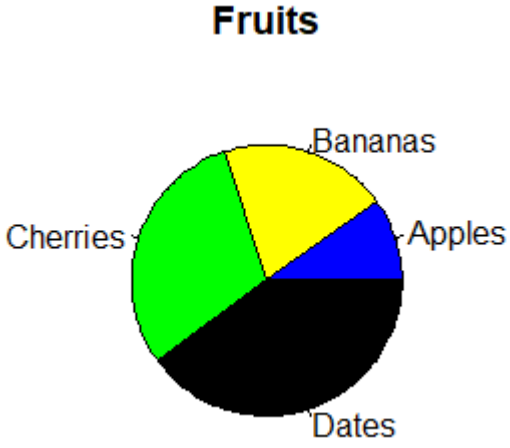


Colors

You can add a color to each pie with the col parameter:

Example

```
> # Create a vector of pies
> x <- c(10,20,30,40)
>
> # Create a vector of labels
> mylabel <- c("Apples", "Bananas", "Cherries", "Dates")
>
> # Create a vector of colors
> colors <- c("blue", "yellow", "green", "black")
>
> # Display the pie chart with labels
> pie(x, label = mylabel, main = "Fruits", col=colors)
```

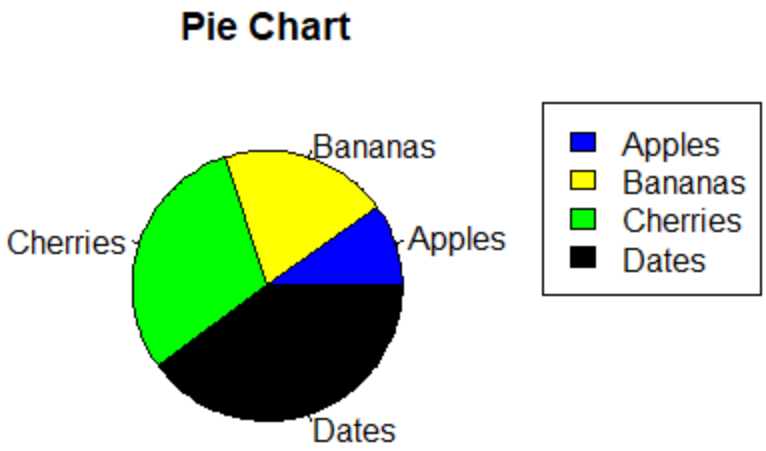


Legend

To add a list of explanation for each pie, use the legend() function:

Example:

```
> # Create a vector of labels
> mylabel <- c("Apples", "Bananas", "Cherries", "Dates")
>
> # Create a vector of colors
> colors <- c("blue", "yellow", "green", "black")
>
> # Display the pie chart with colors
> pie(x, label = mylabel, main = "Pie Chart", col = colors)
>
> # Display the explanation box
> legend("topright", mylabel, fill = colors)
```



The legend can be positioned as either:

bottomright, bottom, bottomleft, left, topleft, top, topright, right, center

R Bar Charts

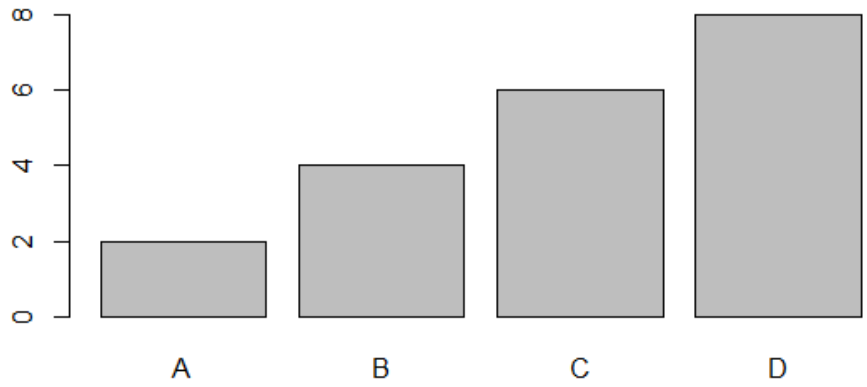
Bar Charts

A bar chart uses rectangular bars to visualize data. Bar charts can be displayed horizontally or vertically. The height or length of the bars are proportional to the values they represent.

Use the barplot() function to draw a vertical bar chart:

Example:

```
> # x-axis values
> x <- c("A", "B", "C", "D")
>
> # y-axis values
> y <- c(2, 4, 6, 8)
>
> barplot(y, names.arg = x)
```



Example Explanation

- The x variable represents values in the x-axis (A,B,C,D)
- The y variable represents values in the y-axis (2,4,6,8)
- Then we use the barplot() function to create a bar chart of the values
- names.arg defines the names of each observation in the x-axis

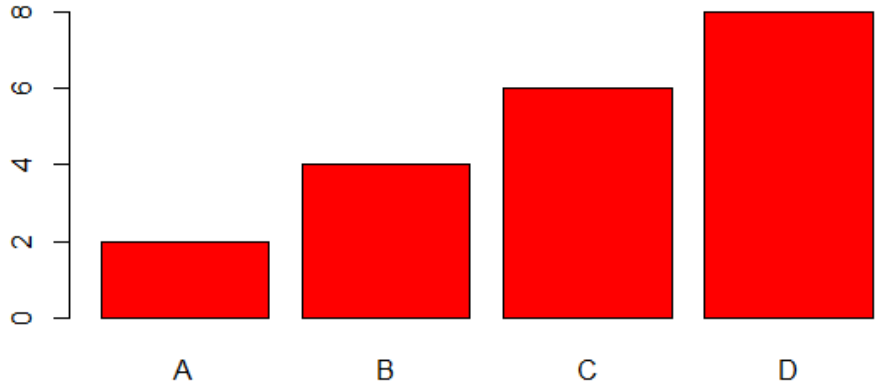
---

Bar Color

Use the col parameter to change the color of the bars:

Example:

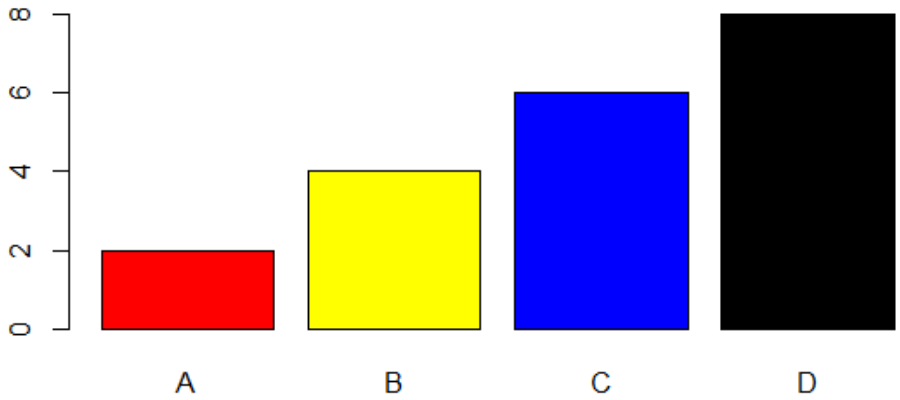
```
> x <- c("A", "B", "C", "D")
> y <- c(2, 4, 6, 8)
>
> barplot(y, names.arg = x, col = "red")
```



You can fill each bar indifferent color:

Example:

```
> # x-axis values
> x <- c("A", "B", "C", "D")
>
> # y-axis values
> y <- c(2, 4, 6, 8)
>
> colors <- c("red", "yellow", "blue", "black")
>
> barplot(y, names.arg = x, col=colors)
```

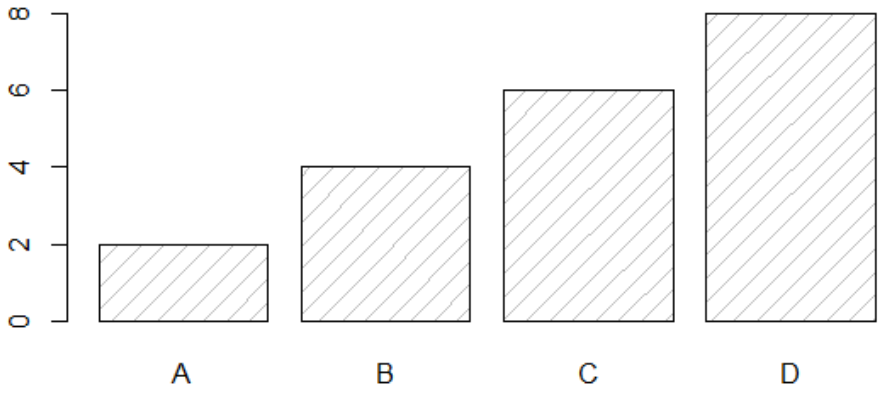


Density / Bar Texture

To change the bar texture, use the density parameter:

Example:

```
> x <- c("A", "B", "C", "D")
> y <- c(2, 4, 6, 8)
>
> barplot(y, names.arg = x, density = 10)
```



Note: the value of density parameter refers to density of the dash lines of texture pattern.

Bar Width

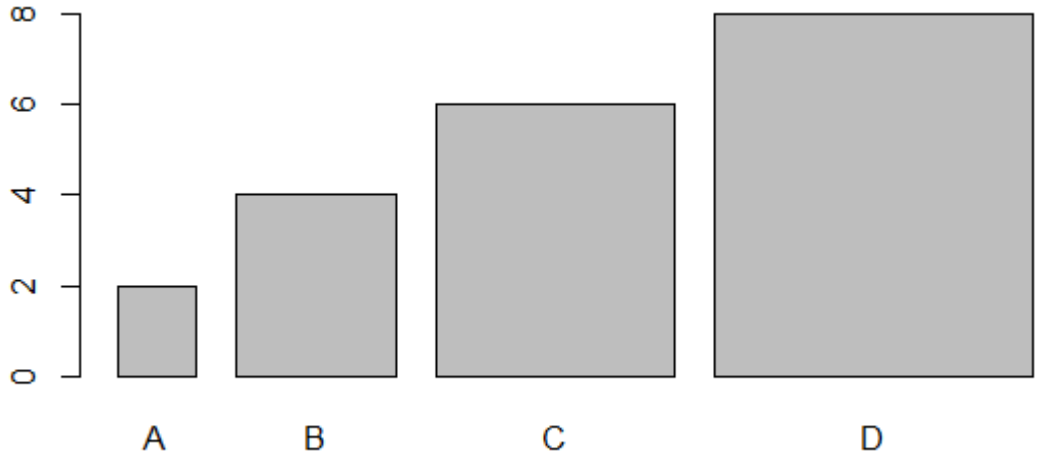
Use the width parameter to change the width of the bars:

Example:

```
> x <- c("A", "B", "C", "D")
> y <- c(2, 4, 6, 8)
>
> barplot(y, names.arg = x, width = c(1,2,3,4))
```

Or could be written:

```
> x <- c("A", "B", "C", "D")
> y <- c(2, 4, 6, 8)
> z <- c(1, 2, 3, 4)
> barplot(y, names.arg = x, width = z)
```

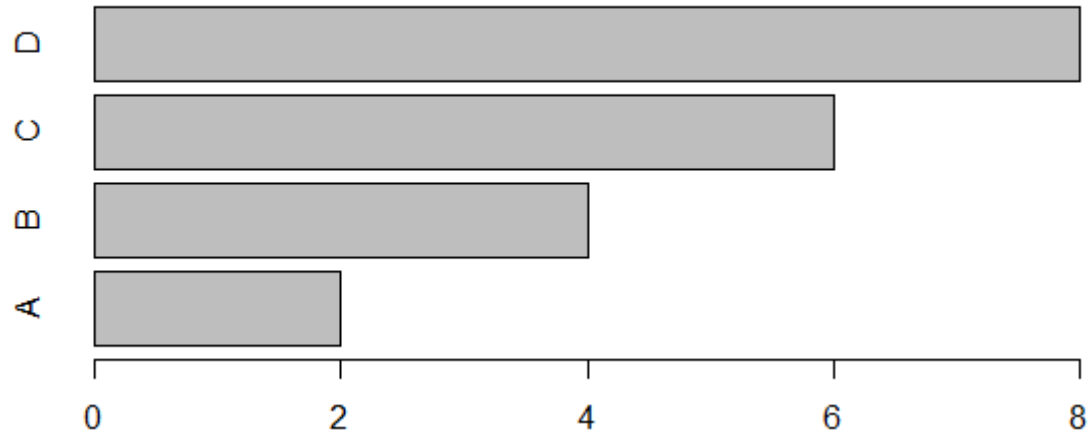


Horizontal Bars

If you want the bars to be displayed horizontally instead of vertically, use horiz=TRUE:

Example:

```
> x <- c("A", "B", "C", "D")
> y <- c(2, 4, 6, 8)
>
> barplot(y, names.arg = x, horiz = TRUE)
```

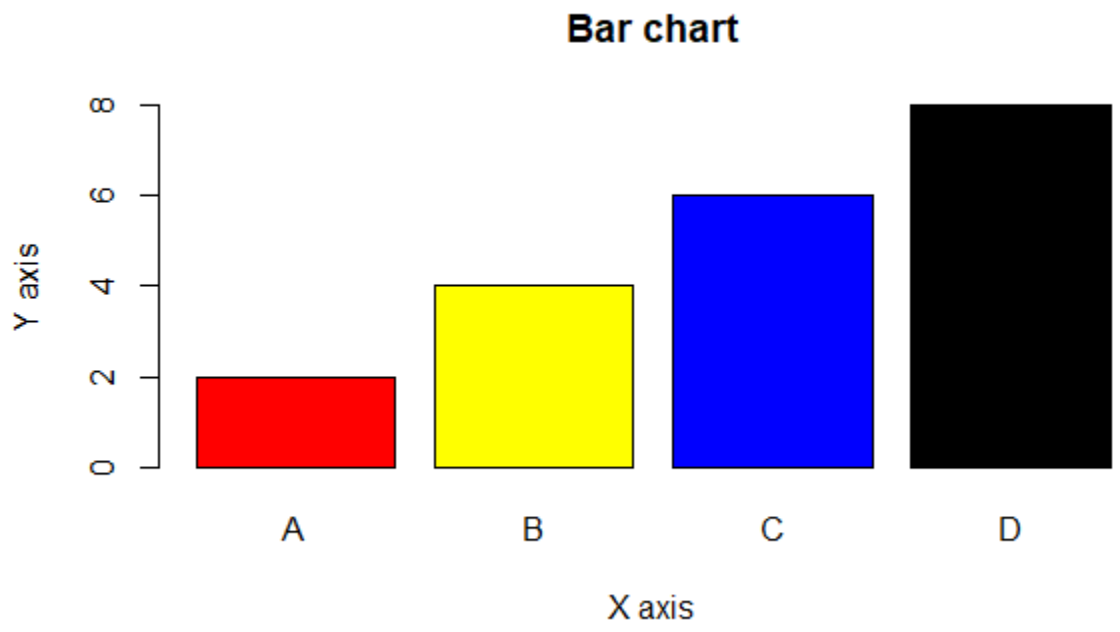


Bar Labels

You can use "main= "Title", xlab= "label", ylab="label" to add labels to the bar chart.

Example:

```
> x <- c("A", "B", "C", "D")
> y <- c(2, 4, 6, 8)
>
> barplot(y, names.arg = x, horiz = TRUE)
> # x-axis values
>  x <- c("A", "B", "C", "D")
>
> # y-axis values
>  y <- c(2, 4, 6, 8)
>
> colors <- c("red", "yellow", "blue", "black")
>
>  barplot(y, names.arg = x, col=colors, main="Bar chart",
xlab="X axis", ylab="Y axis")
```



Exercise 1: Using plot() for Scatter Plots

Problem:

You are given a dataset that contains the height (in cm) and weight (in kg) of 10 individuals. Create a scatter plot to visualize the relationship between height and weight.

Height: (160, 165, 170, 175, 180, 185, 190, 155, 168, 172)

Weight: (55, 60, 65, 70, 75, 80, 85, 50, 58, 72)

==========

Exercise 2: Using barplot() for Categorical Data

Problem:

You are given data on the number of students enrolled in different courses at a university. Create a bar plot to visualize the enrollment numbers.

Dataset:

Courses: ("Math", "Physics", "Chemistry", "Biology", "History")

Enrollment: (120, 90, 80, 110, 70)

==========

Exercise 3: Using pie() for Proportional Data

Problem:

You are given data on the market share of different smartphone brands. Create a pie chart to visualize the market share distribution.

Dataset:

Brands: ("Apple", "Samsung", "Xiaomi", "Huawei", "Others")

market_share: (30, 25, 20, 15, 10)