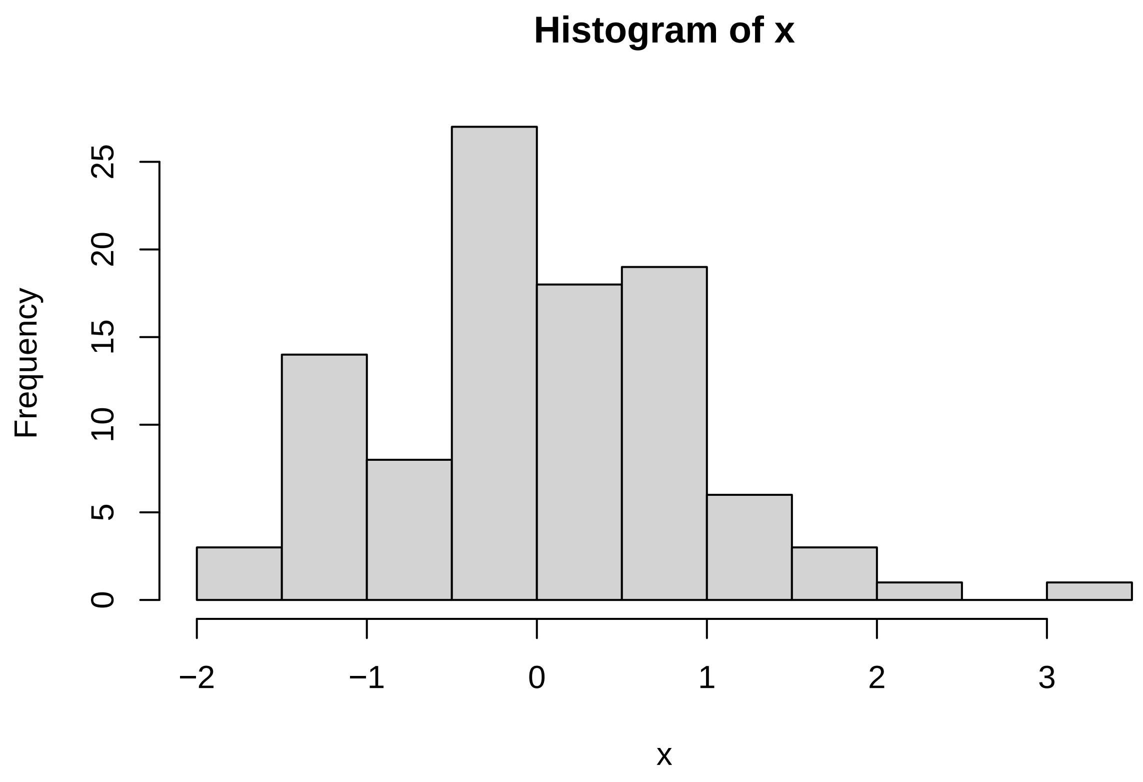
Base Plotting Demonstation

Mr. Sachin B.

24/03/2021

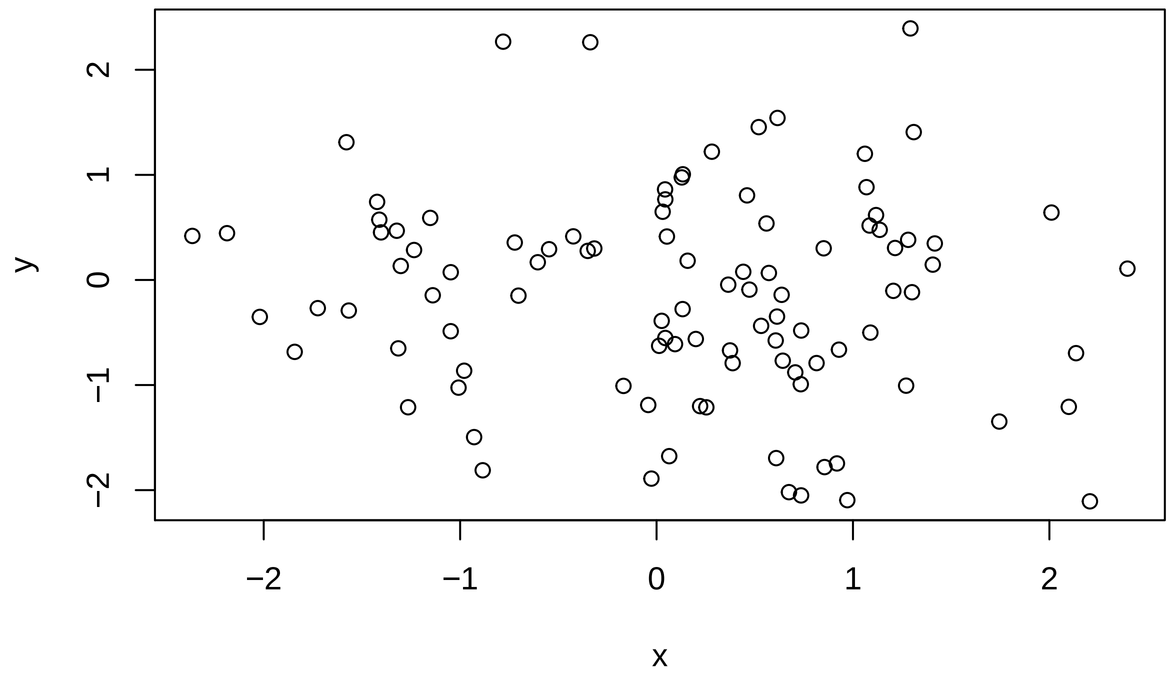
1. Histogram

```r
# 100 random numbers using  normal distribution
x<-rnorm(100)

# Histogram
hist(x)
```

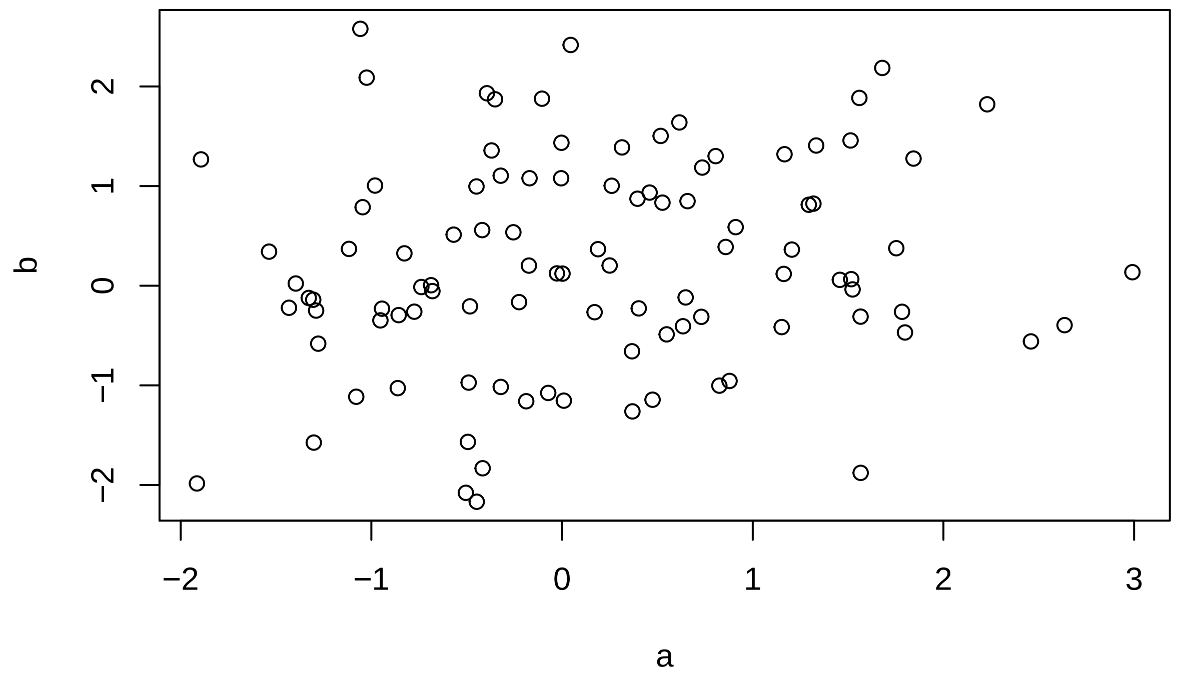


2. Scatter Plot

```r
# 100 random numbers using  normal distribution
x <- rnorm(100)
y <- rnorm(100)

# Scatter Plot
plot(x,y)
```

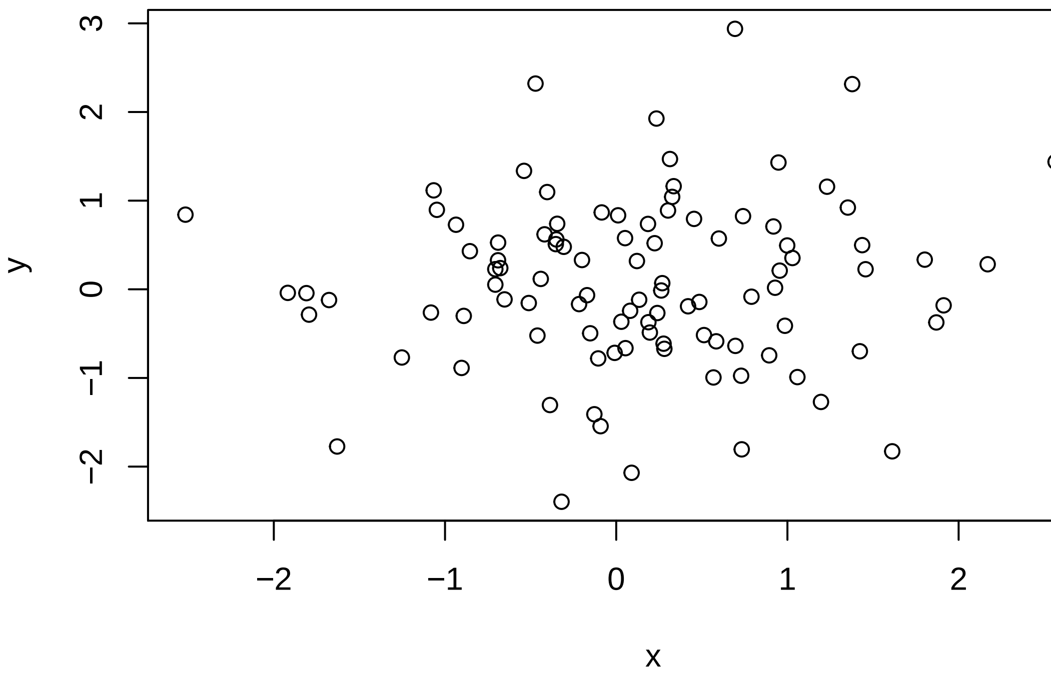


```r
# Scatter Plot Practice
a <- rnorm(100)
b <- rnorm(100)
plot(a,b)
```



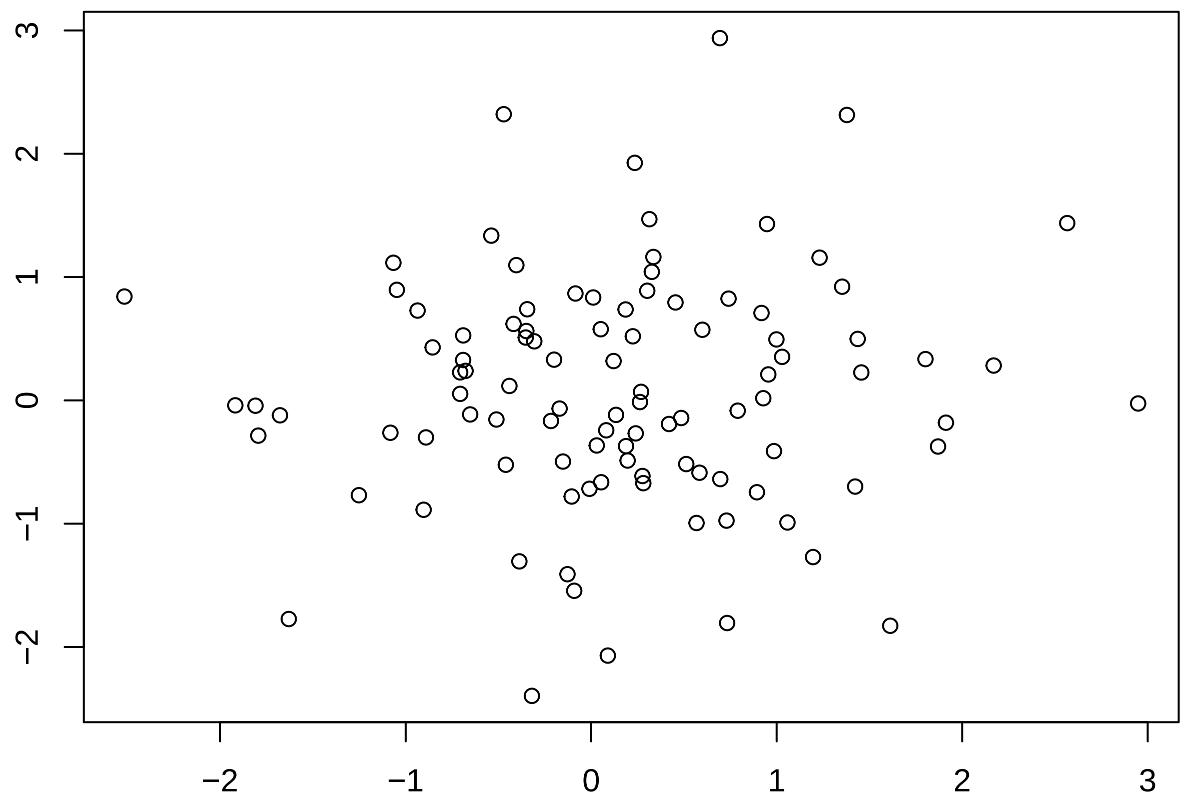
3. Plot Parameter

```r
# 100 random numbers using  normal distribution
x <- rnorm(100)
y <- rnorm(100)

# Scatter Plot Before Applying Margin (Default Margin)
plot(x,y)
```

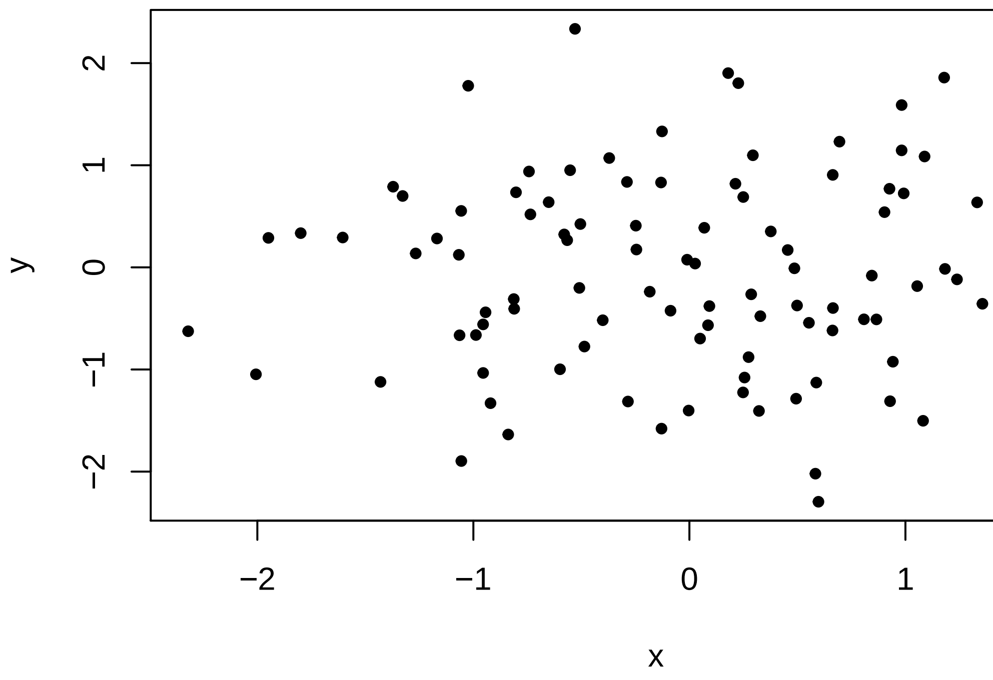


3.1 Plot Parameter: `Margin`

```
# Adding Margin to Scatter Plot
par(mar = c(2,2,2,2))

# Scatter Plot After Applying Margin
plot(x,y)
```

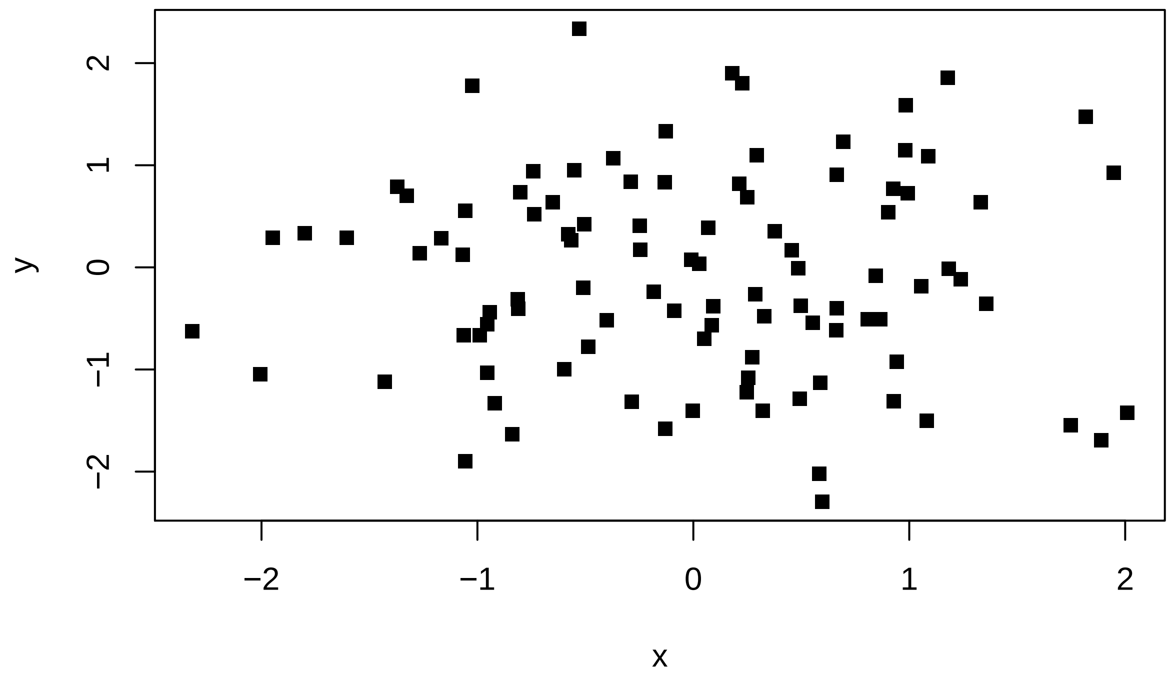


```r
# 100 random numbers using  normal distribution
x <- rnorm(100)
y <- rnorm(100)


# Scatter Plot using pch=20
plot(x,y,pch=20)
```

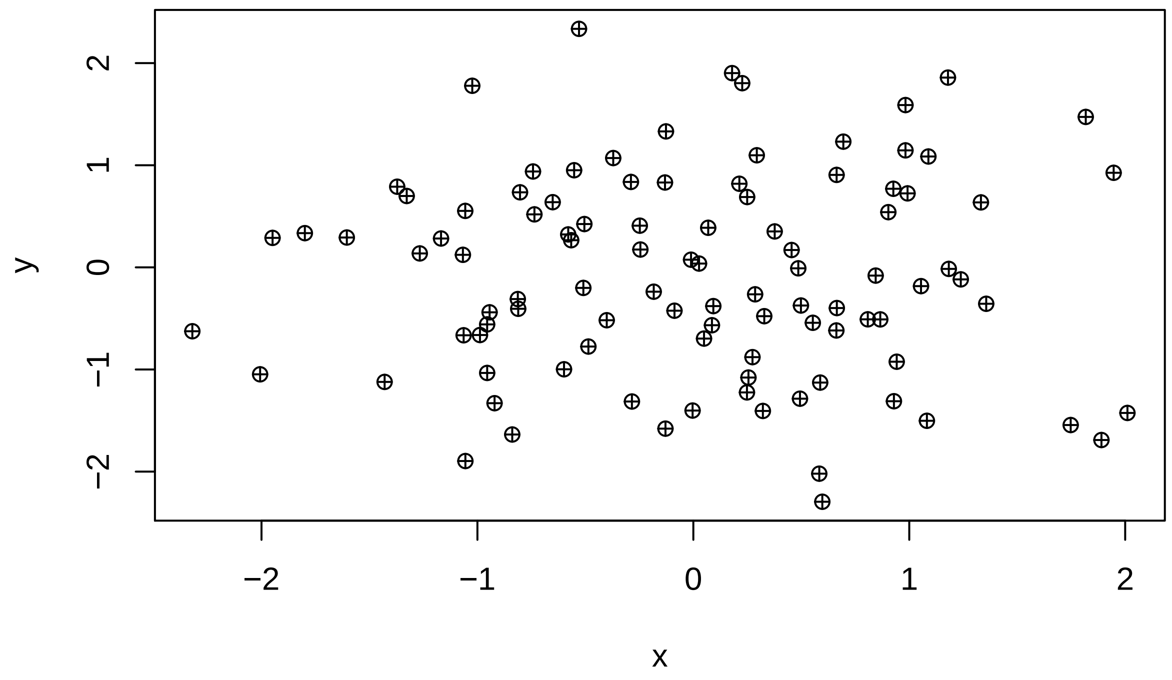


3.2 Scatter Plot Parameter: `pch`

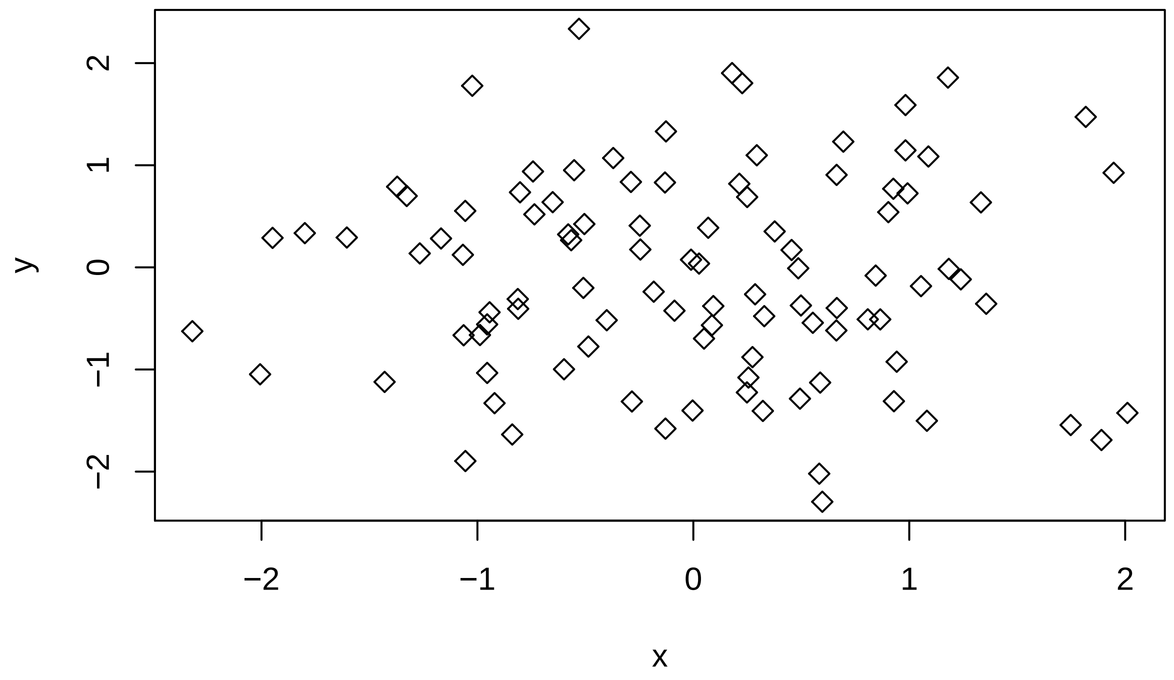
```r
# Scatter Plot using pch=15
plot(x,y,pch=15)
```



```r
# Scatter Plot using pch=10
plot(x,y,pch=10)
```



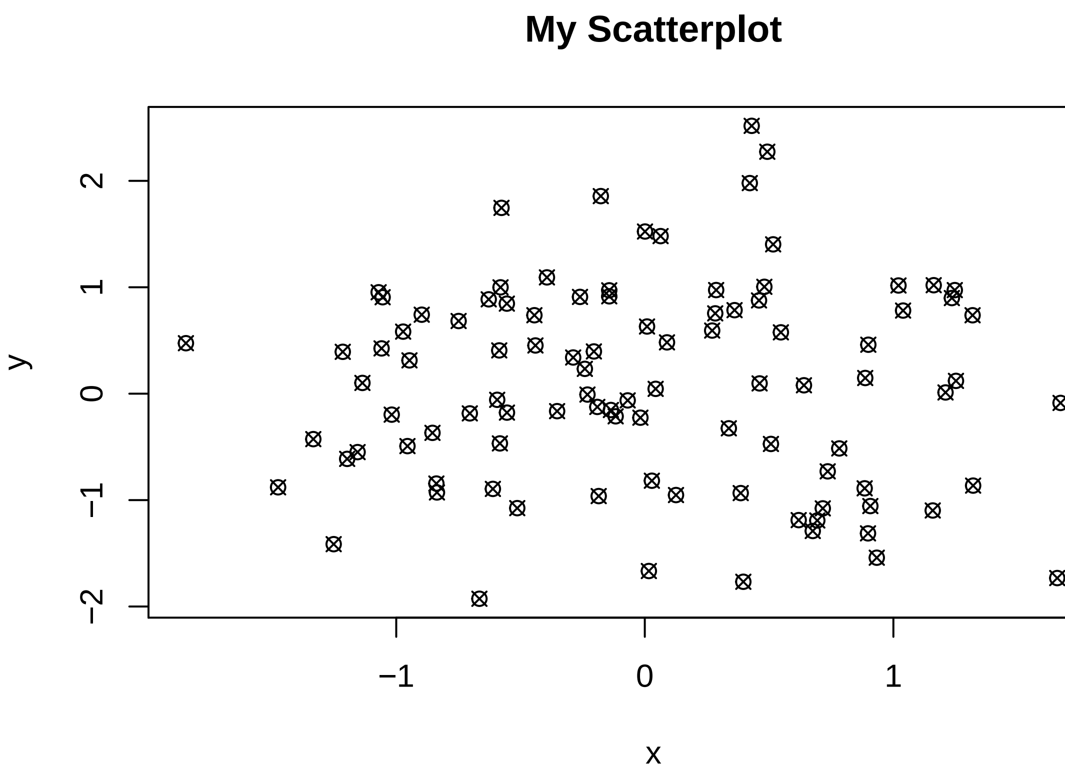
```
# Scatter Plot using pch=5
plot(x,y,pch=5)
```



```r
example("points")
```

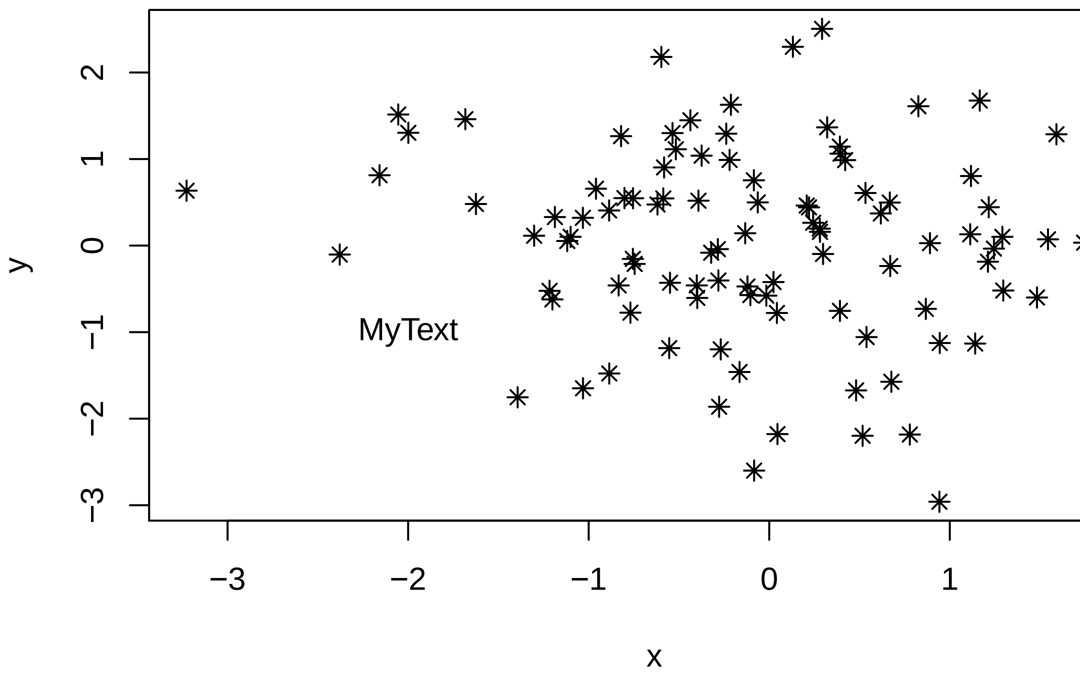
To know more about pch symbol

```r
# 100 random numbers using  normal distribution
x <- rnorm(100)
y <- rnorm(100)

# Scatter Plot
plot(x,y,pch=13)

# Adding Title to Plot
title("My Scatterplot")
```



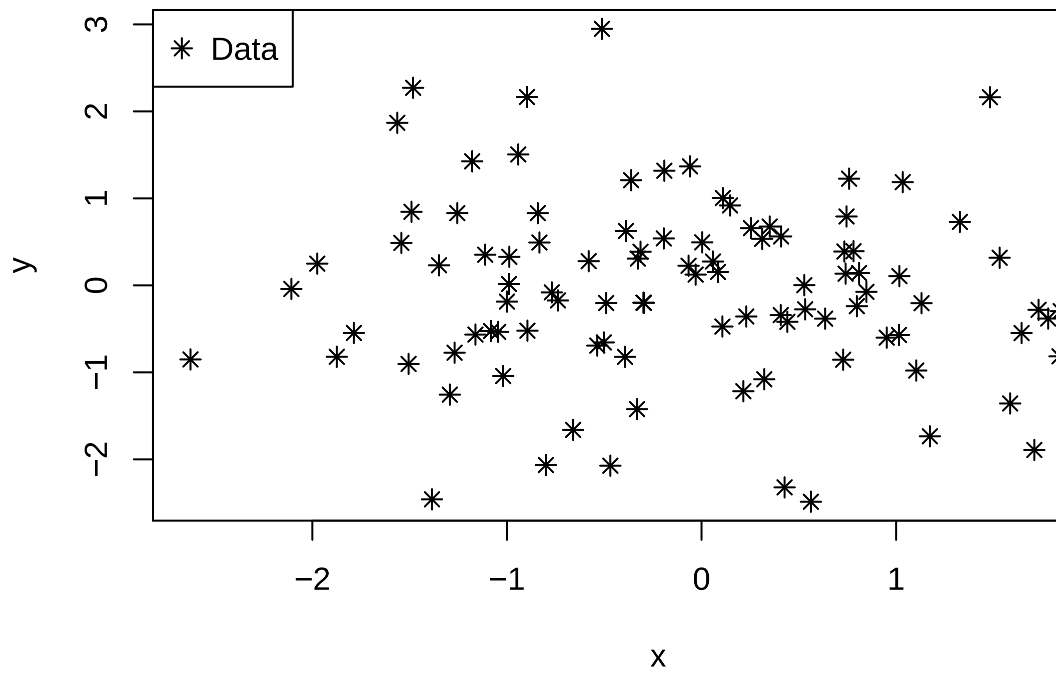
3.3 Plot Parameter: `title`

```r
# 100 random numbers using  normal distribution
x <- rnorm(100)
y <- rnorm(100)

# Scatter Plot
plot(x,y,pch=8)

# Adding Text in a Plot
text(-2,-1,"MyText")
```



3.4 Plot Parameter: `text`

```
# 100 random numbers using  normal distribution
x <- rnorm(100)
y <- rnorm(100)

# Scatter Plot
plot(x,y,pch=8)

# Adding Legend to a Plot
legend("topleft",legend = "Data", pch = 8)
```



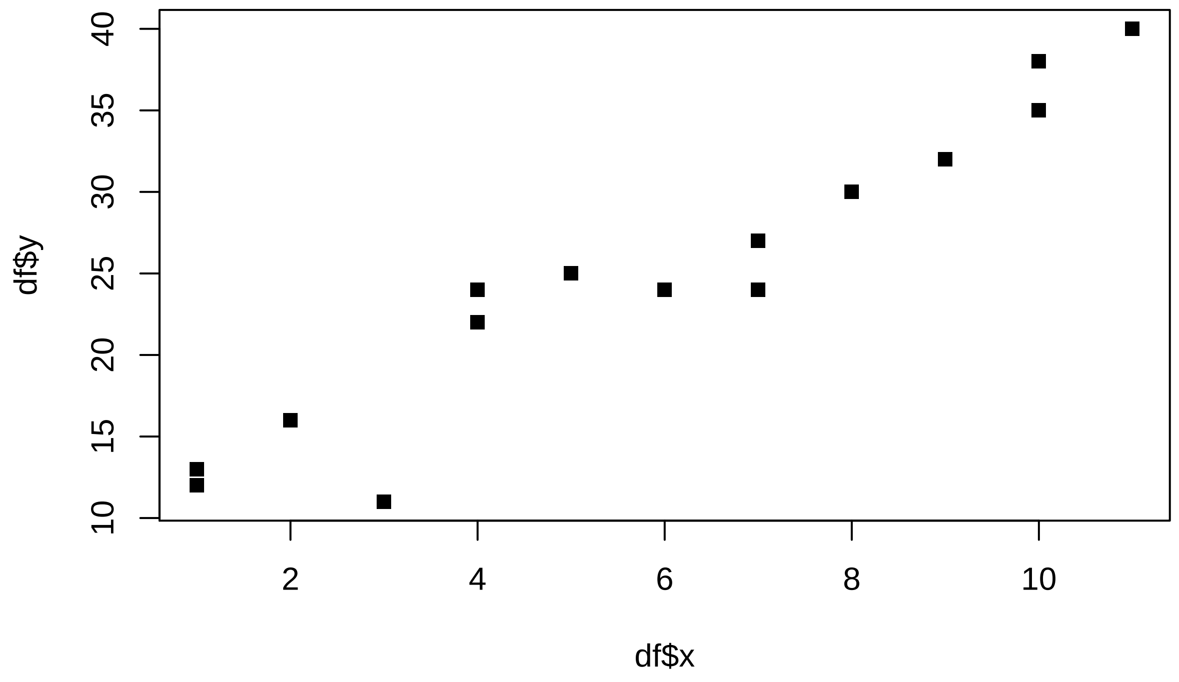
3.5 Plot Parameter: `legend`

3.6 Plot Parameter: `abline` `abline` function adds one or more straight lines through the current plot. The basic syntax is of abline() is as follows:

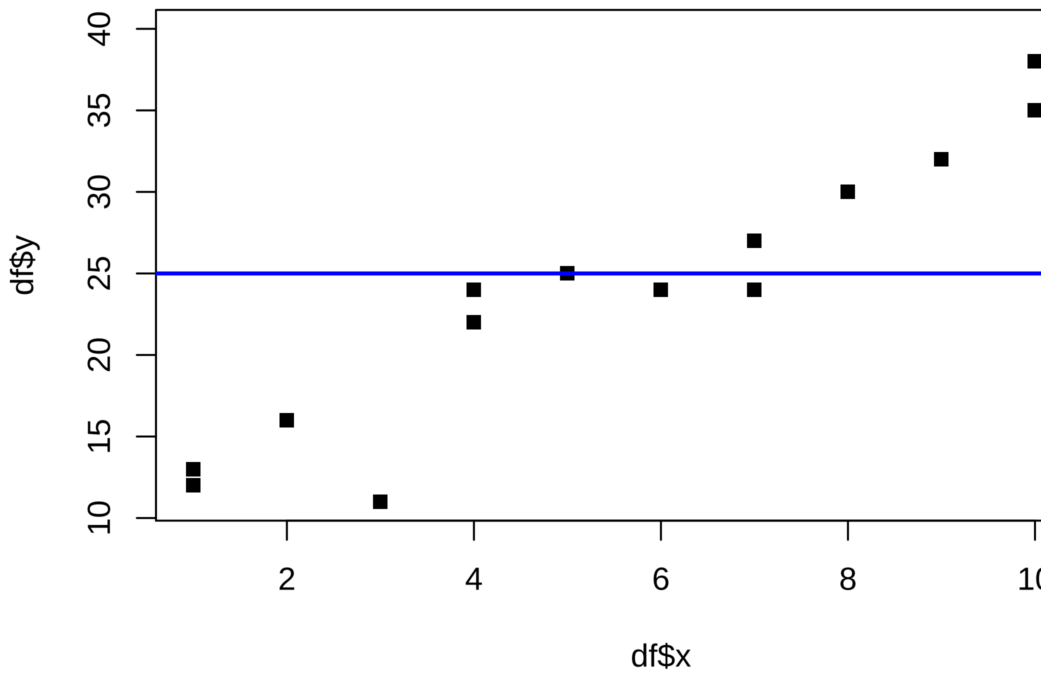
```
abline(a=NULL, b=NULL, h=NULL, v=NULL, ...)
```

- a, b: single values that specify the intercept `(a)` and slope `(b)` of the line
- h: the y-value for the horizontal line
- v: the x-value for the vertical line

```
#define dataset

df <- data.frame(x = c(1,1,2,3,4,4,5,6,7,7,8,9,10,10,11),
                 y = c(12,13,16,11,22,24,25,24,24,27,30,32,35,38,40))

#plot x and y values in dataset
plot(df$x, df$y, pch = 15)
```



```r
# To add a horizontal line at the value y = 25
plot(df$x, df$y, pch = 15)

abline(h = 25, col = 'blue', lwd = 2)
```

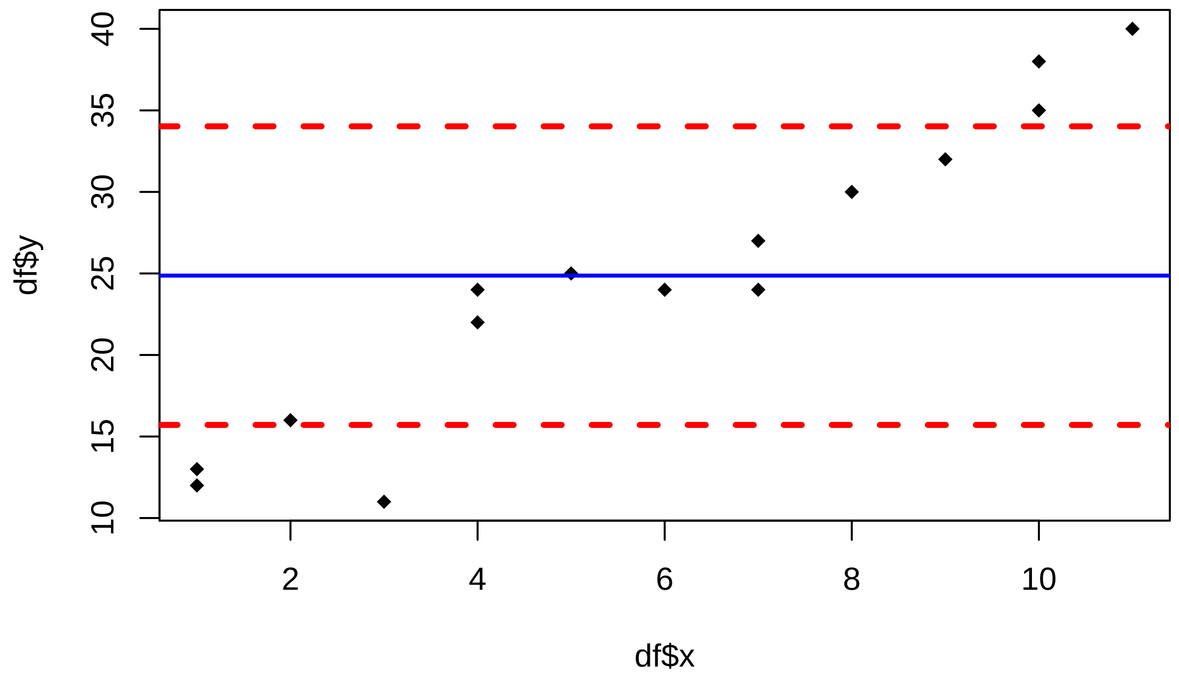


3.6.1 `abline`:`Horizontal Line`

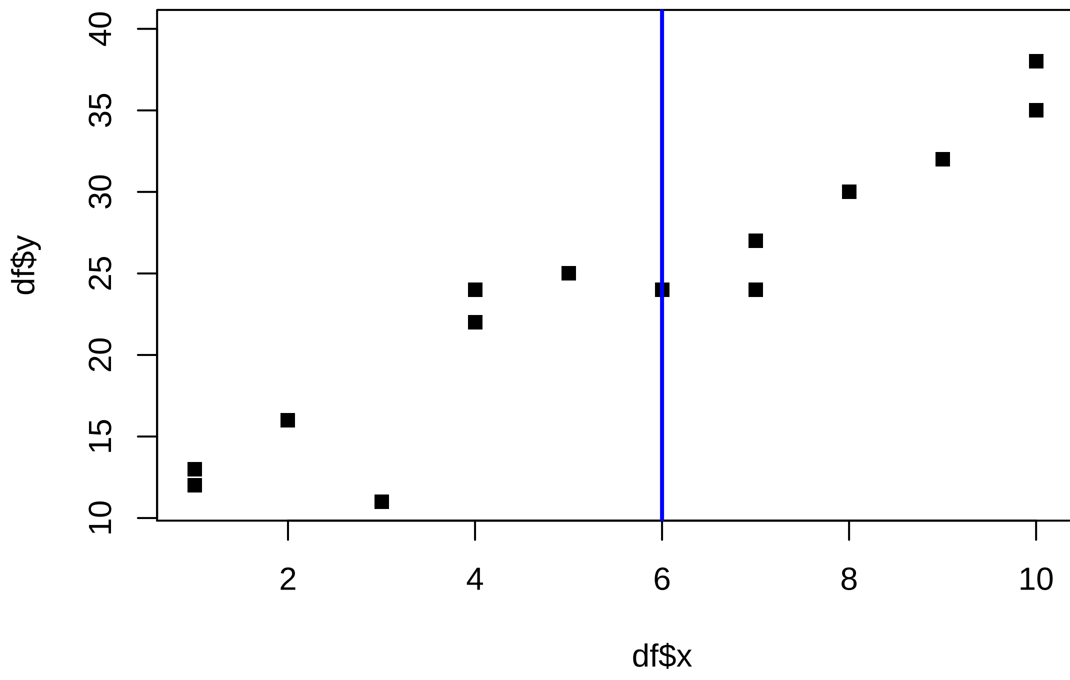
The following code illustrates how to add a horizontal solid line at the mean value of y along with two horizontal dashed lines at one standard deviation above and below the mean value

- Note that lwd = 2 specifies that we want the line width to be equal to 2 (default = 1).
- Note that lty = 2 specifies that we want the line to be dashed.

```
#create scatterplot for x and y
plot(df$x, df$y, pch = 18)

#create horizontal line at mean value of y
abline(h = mean(df$y), col='blue', lwd = 2)

#create horizontal lines at one standard deviation above and below the mean value
abline(h = mean(df$y) + sd(df$y), col = 'red', lwd = 3, lty = 2)
abline(h = mean(df$y) - sd(df$y), col = 'red', lwd = 3, lty = 2)
```



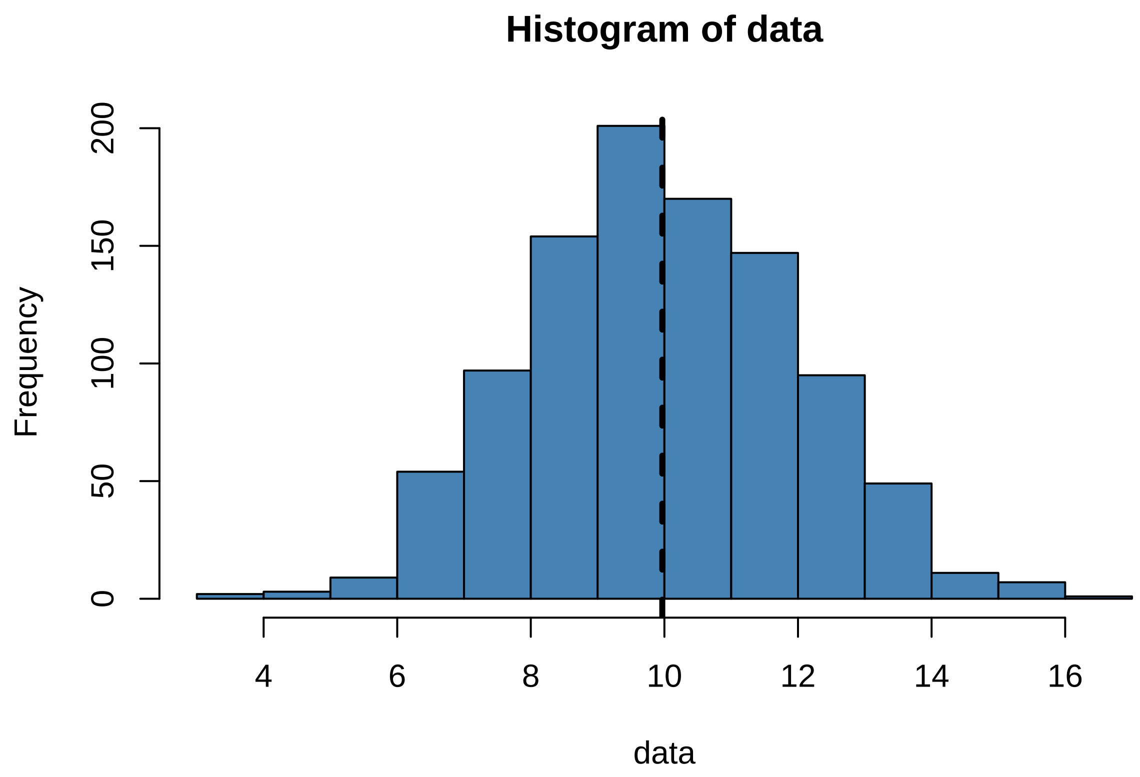
```r
# To add a vertical line at the value x = 6
plot(df$x, df$y, pch = 15)

abline(v = 6, col = 'blue', lwd = 2)
```



3.6.2 `abline:Vertical Line`

The following code illustrates how to add a vertical line at the mean value on a histogram

```
#make this example reproducible
set.seed(0)

#create dataset with 1000 random values normally distributed with mean = 10, sd = 2
data <- rnorm(1000, mean = 10, sd = 2)

#create histogram of data values
hist(data, col = 'steelblue')

#draw a vertical dashed line at the mean value
abline(v = mean(data), lwd = 3, lty = 2)
```

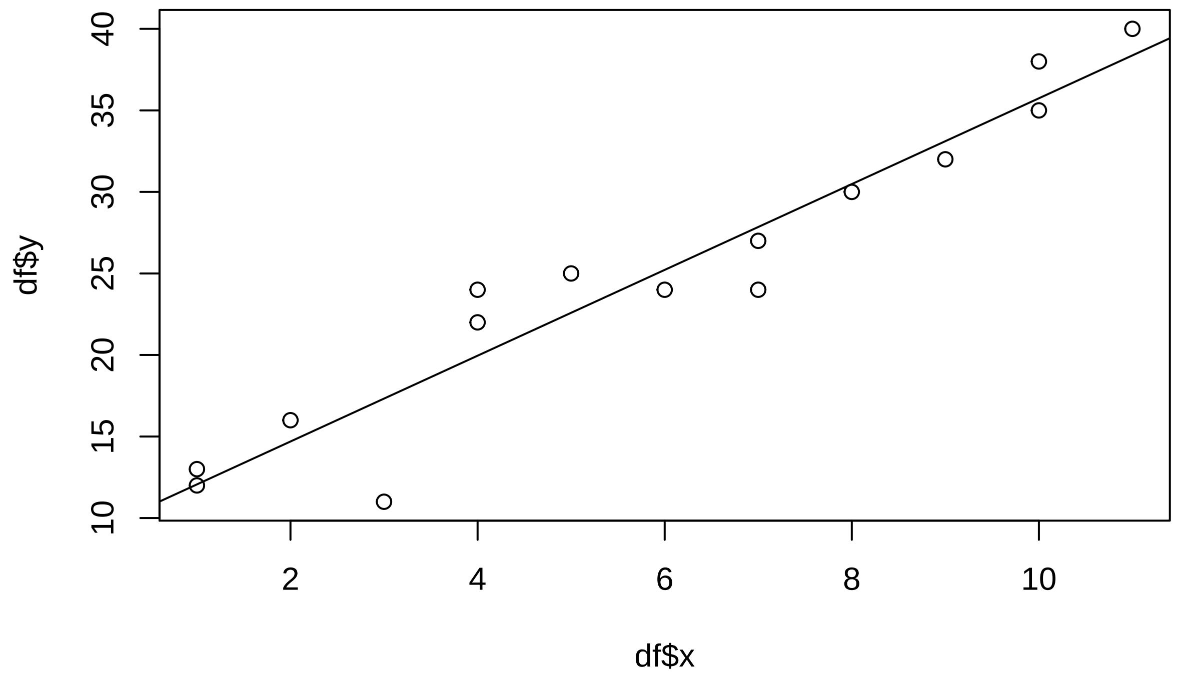


3.6.3 `abline:Regression Line` The basic code to add a simple linear regression line to a plot in R is: `abline(reg_model)`

- where `reg_model` is a fitted regression line created by using the `lm()` function.

The following code illustrates how to add a fitted linear regression line to a scatterplot

```
#create scatterplot of x and y values
plot(df$x, df$y)

#fit a linear regression model to the data
reg_model <- lm(y ~ x, data = df)

#add the fitted regression line to the scatterplot
abline(reg_model)
```



Note that we simply need a value for the intercept and the slope to fit a simple linear regression line to the data using the abline() function.

Thus, another way (although a more tedious way) of using abline() to add a regression line is to explicitly specify the intercept and slope coefficients of the regression model:

```r
#define dataset
df <- data.frame(x = c(1,1,2,3,4,4,5,6,7,7,8,9,10,10,11),
                 y = c(12,13,16,11,22,24,25,24,24,27,30,32,35,38,40))

#plot x and y values in dataset
plot(df$x, df$y, xlim = c(0,10), ylim = c(0,40))

#fit a linear regression model to the data
reg_model <- lm(y ~ x, data = df)
print(reg_model)
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Coefficients:
## (Intercept)            x
##       9.432        2.631
```

```r
print(class(reg_model))
```

```
## [1] "lm"
```

```r
print(typeof(reg_model))
```

```
## [1] "list"
```

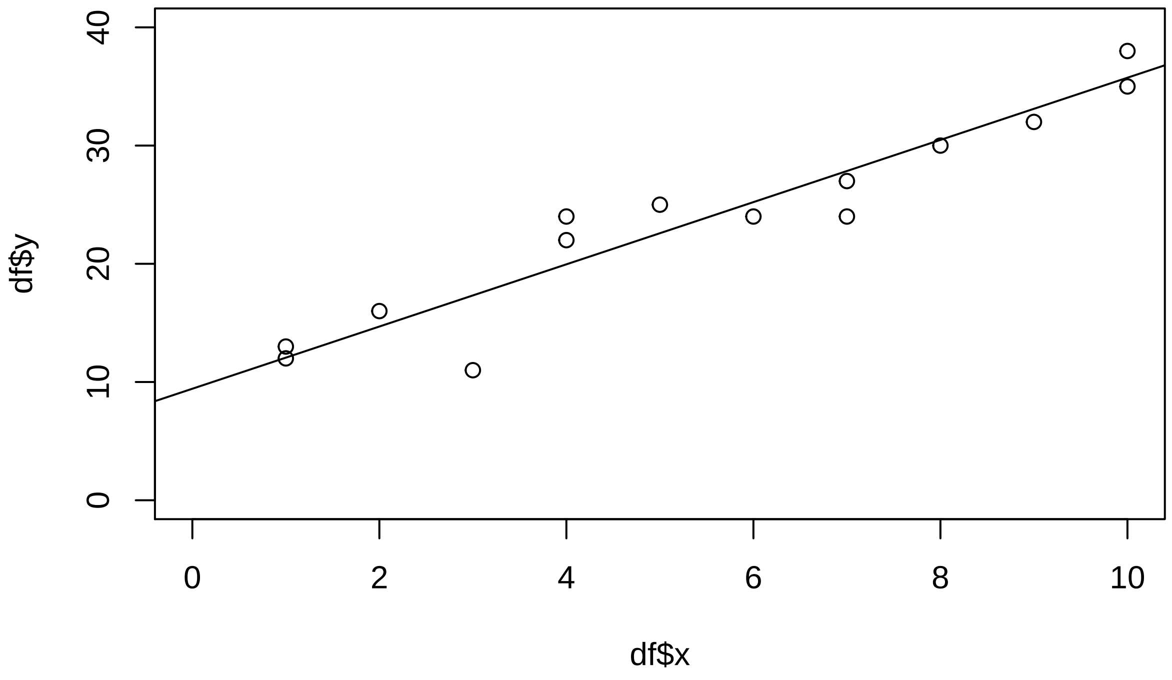
```r
#define intercept and slope values
a <- coefficients(reg_model)[1] #intercept
b <- coefficients(reg_model)[2] #slope

print(a)
```

```
## (Intercept)
##    9.431507
```

```r
print(b)
```

```
##        x
## 2.630993
```

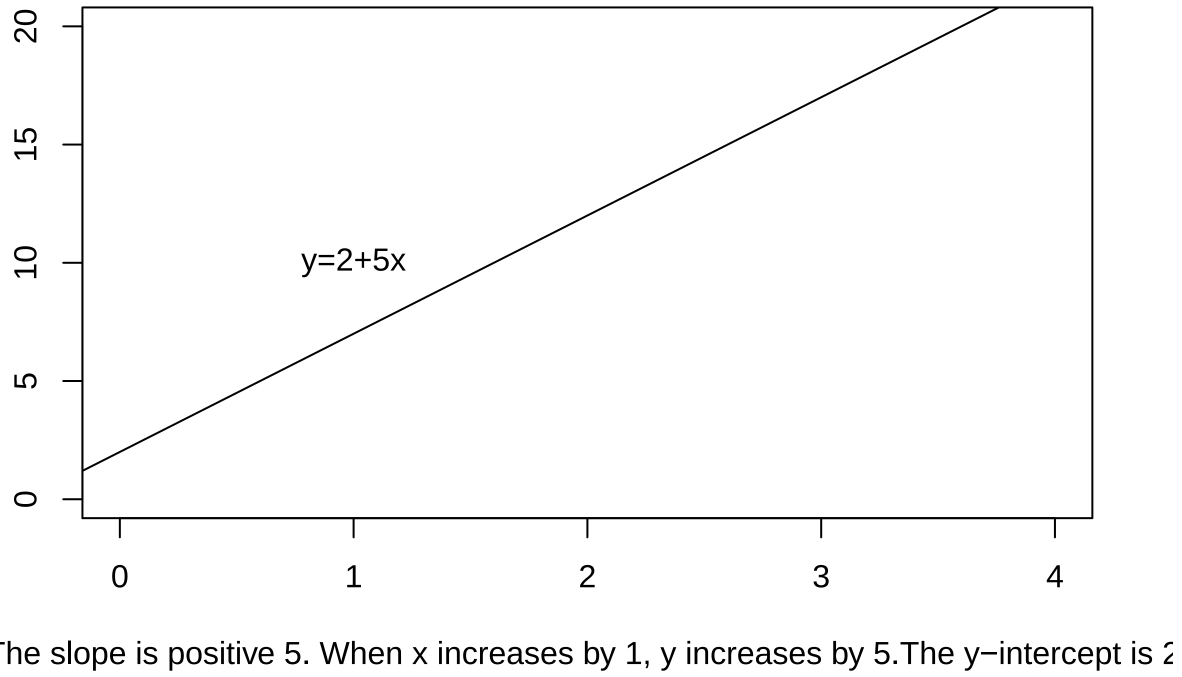
```r
#add the fitted regression line to the scatterplot
abline(a=a, b=b)
```



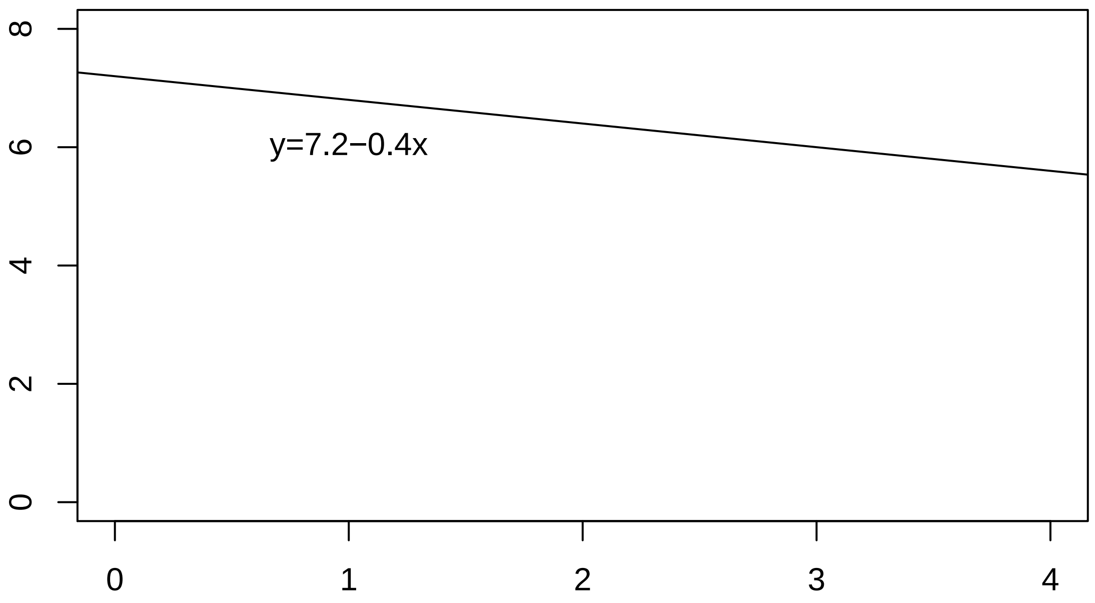
Slope and intercept of the regression line

- Slope indicates the steepness of a line
- Intercept indicates the location where it intersects an axis
- The slope and the intercept define the linear relationship between two variables, and can be used to estimate an average rate of change.
- The greater the magnitude of the slope, the steeper the line and the greater the rate of change.

```r
# y=2+5x
plot(1,type = "n", xlab = "The slope is positive 5. When x increases by 1, y increases by 5.The y-inter

text(1,10,"y=2+5x")
abline(a=2, b=5)
```

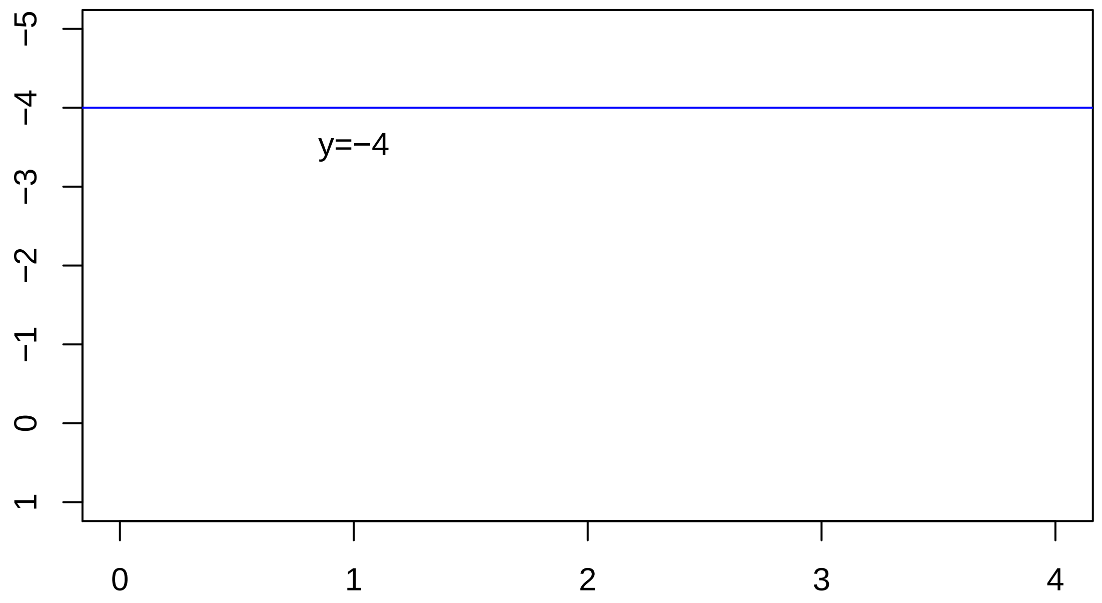


```r
# y=7.2-0.4x
plot(1,type = "n", xlab = "The slope is negative 0.4. When x increases by 1, y decreases by 0.4. The y-

text(1,6,"y=7.2-0.4x")
abline(a=7.2, b=-0.4)
```



The slope is negative 0.4. When x increases by 1, y decreases by 0.4. The y−intercept is

```
# y=-4
plot(1,type = "n", xlab = "The slope is 0. When x increases by 1, y neither increases/decreases. The y-i

text(1,-3.5,"y=-4")
abline(a=-4, b=0, col="blue")
```



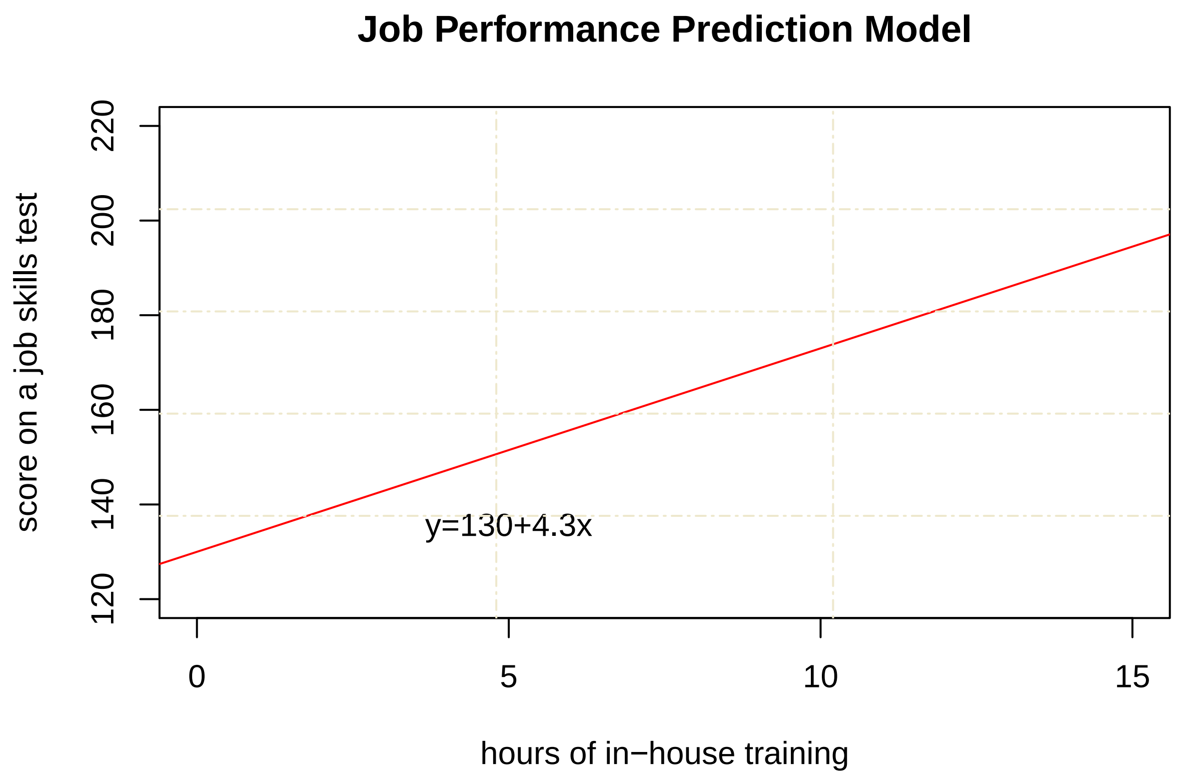
The slope is 0. When x increases by 1, y neither increases/decreases. The y−intercept i

Usually, this relationship can be represented by the equation $y = b_0 + b_1x$, where b_0 is the $y - intercept$ and b_1 is the *slope*.

Example A company determines that job performance for employees in a production department can be predicted using the regression model $y = 130 + 4.3x$ where,

- x is the hours of in-house training they receive (from 0 to 20)
- y is their score on a job skills test.
- The value of the $y - intercept$ (130) indicates the average job skill score for an employee with no training.
- The value of the *slope* (4.3) indicates that for each hour of training, the job skill score increases, on average, by 4.3 points.

```
# y=130+4.3x
plot(1,type = "n", xlab = "hours of in-house training",ylab = "score on a job skills test",  xlim = c(0

title("Job Performance Prediction Model")

text(5,135,"y=130+4.3x")
abline(a=130, b=4.3, col="red")
grid (3,5, lty = 6, col = "cornsilk2")
```

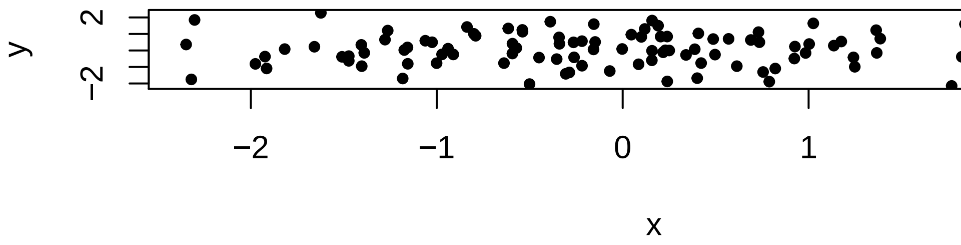


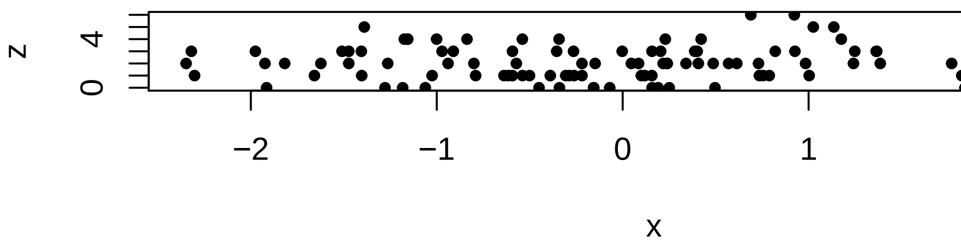
3.7 Multiple Plot in Single Canvas

- `mfrow()`
- `mfcol()`

```r
# 100 random numbers
x <- rnorm(100)
y <- rnorm(100)
z <- rpois(100,2)
```

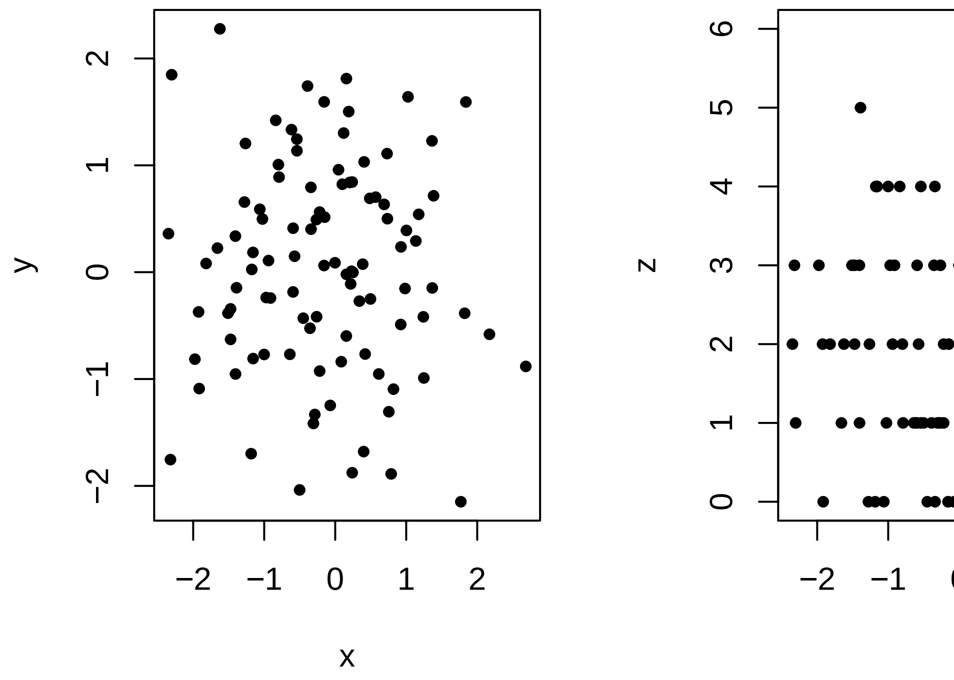
```r
par(mfrow = c(2,1))
plot(x,y,pch=20)
plot(x,z,pch=20)
```





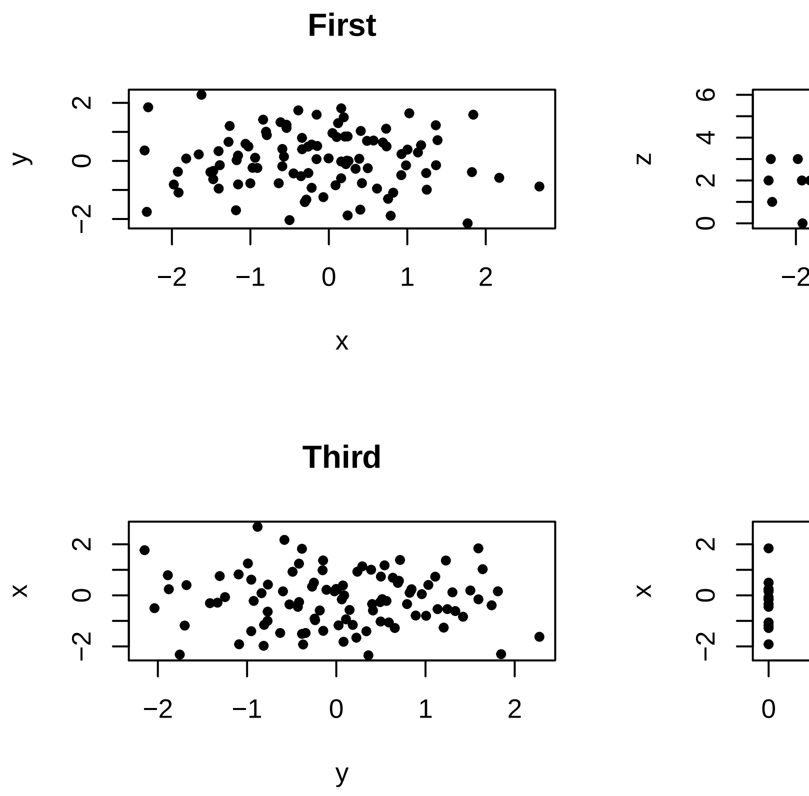
Multiple Plot: 2-rows & 1-column

```
par(mfrow = c(1,2))
plot(x,y,pch=20)
plot(x,z,pch=20)
```



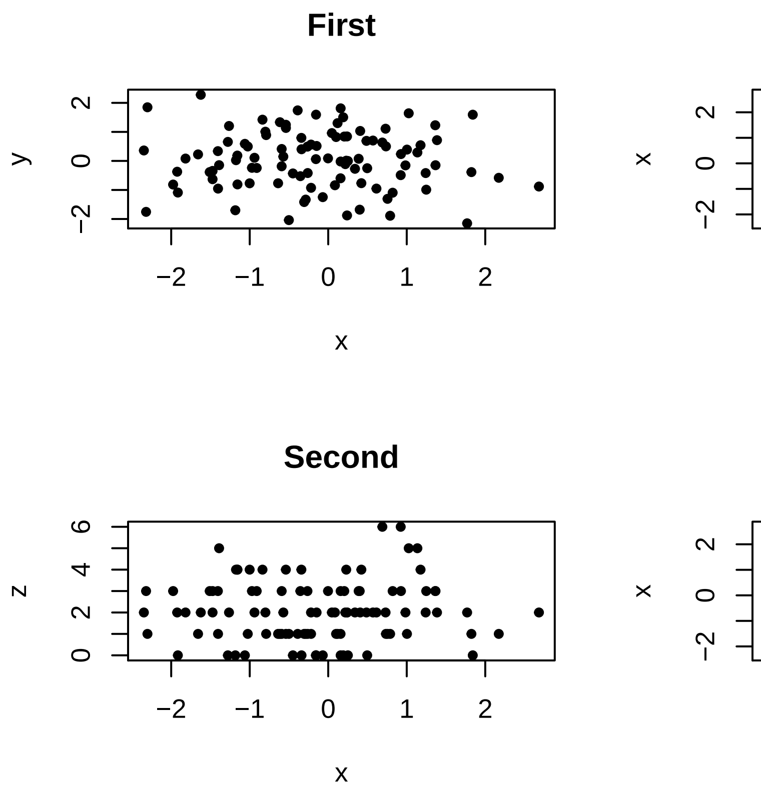
Multiple Plot: 1-rows & 2-columns

```
par(mfrow = c(2,2))
plot(x,y,pch=20, main = "First")
plot(x,z,pch=20, main = "Second")
plot(y,x,pch=20, main = "Third")
plot(z,x,pch=20, main = "Forth")
```



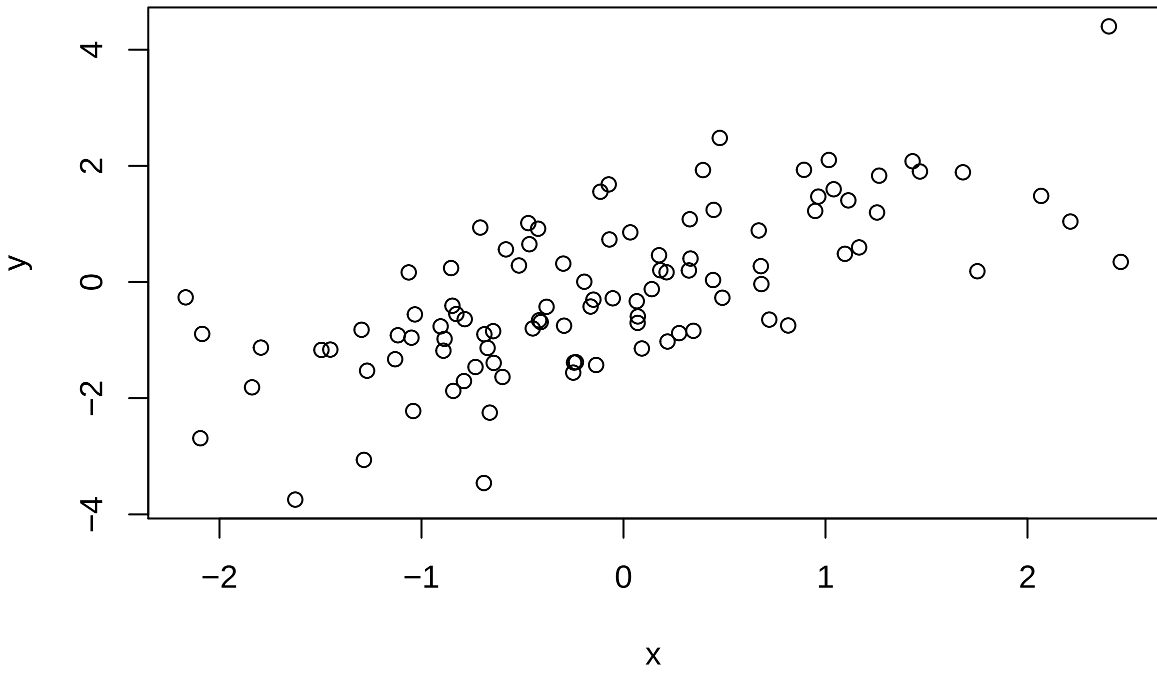
Multiple Plot (row-wise): 2-rows & 2-columns

```
par(mfcol = c(2,2))
plot(x,y,pch=20, main = "First")
plot(x,z,pch=20, main = "Second")
plot(y,x,pch=20, main = "Third")
plot(z,x,pch=20, main = "Forth")
```



Multiple Plot (column-wise): 2-rows & 2-columns

```
# Plot of random numbers
x <- rnorm(100)
y <- x + rnorm(100)

plot(x,y)
```

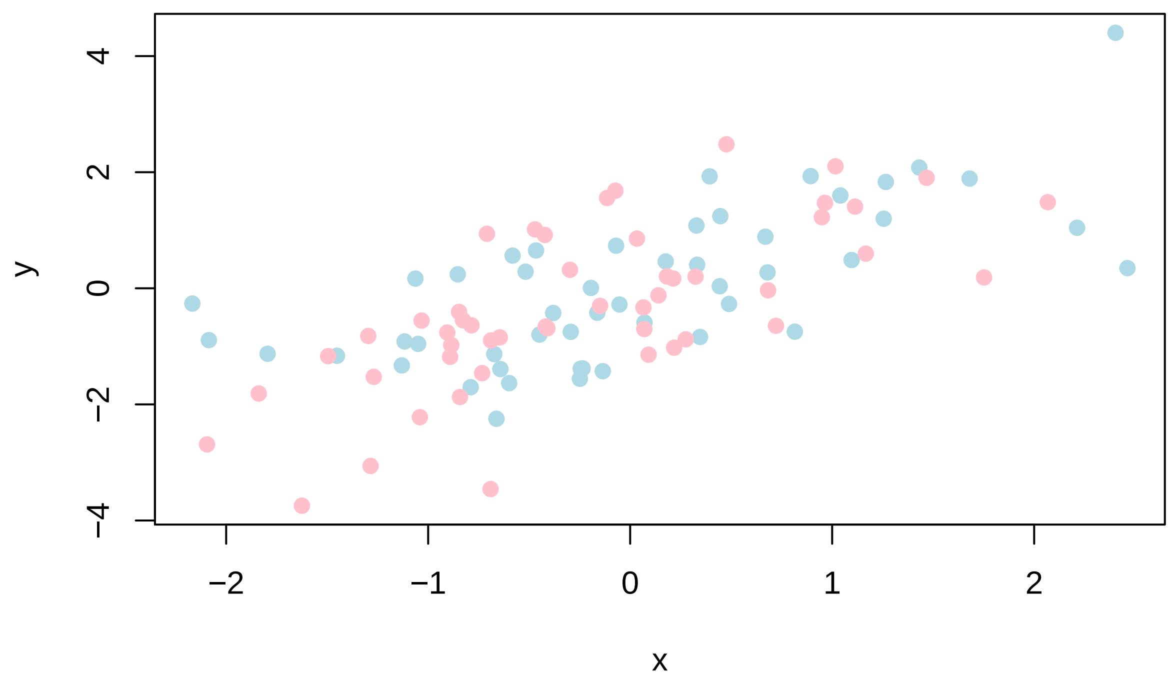


3.8 Points Variable

```
# Factor Object containing Male and Female data

g <- gl(2,50, labels = c("Male","Female"))

print(g)
```

```
##   [1] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
##  [11] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
##  [21] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
##  [31] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
##  [41] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
##  [51] Female Female Female Female Female Female Female Female Female Female
##  [61] Female Female Female Female Female Female Female Female Female Female
##  [71] Female Female Female Female Female Female Female Female Female Female
##  [81] Female Female Female Female Female Female Female Female Female Female
##  [91] Female Female Female Female Female Female Female Female Female Female
## Levels: Male Female
```

```
# Blank Plot
plot(x,y, type = "n")

# Add Male points to graph
points(x[g=="Male"],y[g=="Male"], col = "light blue",pch=19)

# Add Female points to graph
points(x[g=="Female"],y[g=="Female"], col = "pink", pch=19)
```



y
4
2
0
−2
−4
−2
−1
0
1
2
x