**Writing to a File**

kids <- c("Jack","Jill")

ages <- c(12,10)

d <- data.frame(kids,ages,stringsAsFactors=FALSE)

d

write.table(d,"kds")

write.table(xc,"xcnew",row.names=FALSE,col.names=FALSE)

cat("abc\n",file="u")

cat("de\n",file="u",append=TRUE)

cat(file="v",1,2,"xyz\n")

c <- file("www","w")

writeLines(c("abc","de","f"),c)

close(c)

?files

**String-Manipulation Functions**

**grep()**

grep(pattern,x)

grep("hi",c("csd","csd hi","hi csd"))

**nchar()**

nchar("hello csd")

**paste()**

paste("hello csd ")

paste("hello csd ","hi",sep="")

paste("hello csd ","hi",sep=".")

paste("hello csd ","and"," hi ","csm")

**sprintf()**

i <- 8

s <- sprintf("the square of %d is %d",i,i^2)

s

**substr()**

substr(x,start,stop) hello r pgm

substring("csdhiwelcome",3,5)

**strsplit()**

strsplit(x,split)

strsplit("6-16-2011",split="-")

**regexpr()**

regexpr(pattern,text)

regexpr("dhi"," csdhiwelcome")

**gregexpr()**

gregexpr(pattern,text)

gregexpr("hihi","hicsdhihi")

**Regular Expressions**

grep("[hi]",c("csd","csd hi","hi csd"))

grep("c.s",c("csd","csd hi","hi csd"))

grep(".",c("abc","de","f.g"))

grep("\\.",c("abc","de","f.g"))

**Creating in a notepad and saving a file with .txt**

123

4 5

6

scan("z1.txt")

123

4.2 5

6

scan("z2.txt")

abc

de f

g

scan("z3.txt")

scan("z3.txt",what="")

abc

123 6

y

scan("z4.txt",what="")

v <- scan("z1.txt")

x1 <- scan("z3.txt",what="")

x2 <- scan("z3.txt",what="",sep="\n")

x1

x2

x1[2]

x2[2]

v <- scan("")

v

**Using the readline() Function**

w <- readline()

w

inits <- readline("type your initials: ")

type your initials: NM

inits

**Printing to the Screen**

x <- 1:3

print(x^2)

print("abc")

cat("abc\n")

x

cat(x,"abc","de\n")

cat(x,"abc","de\n",sep="")

cat(x,"abc","de\n",sep="\n")

x <- c(5,12,13,8,88)

cat(x,sep=c(".",".",".","\n","\n"))

**Reading and Writing Files**

name age

John 25

Mary 28

Jim 19

z <- read.table("z",header=TRUE)

z

**Reading Text Files**

z1 <- readLines("z1")

z1

**Introduction to Connections**

c <- file("z1.txt","r")

readLines(c,n=1)

readLines(c,n=1)

readLines(c,n=1)

readLines(c,n=1)

**(OR)**

**c <- file("z1.txt","r")**

**while(TRUE) {**

**rl <- readLines(c,n=1)**

**if (length(rl) == 0) {**

**print("reached the end")**

**break**

**} else print(rl)**

**}**

c <- file("z1.txt","r")

readLines(c,n=2)

seek(con=c,where=0)

readLines(c,n=1)

**Creating Graphs**

**Creating graphs:**

Plot () then we’ll explore how to build a graph, from adding lines and points to attaching a legend.

**The Workhorse of R Base Graphics: The plot () Function:**

When we call plot () with an X vector and a Y vector, which are interpreted as a set of pairs in the (x,y) plane.

Plot(c (1, 2, 3), c (1, 2, 4))

plotting the points (1,1), (2,2), and (3,4)

plot(c(-3,3), c(-1,5), type = "n", xlab="x", ylab="y")

This draws axes labeled x and y. The horizontal (x) axis ranges from −3 to 3. The vertical (y) axis ranges from −1 to 5. The argument type="n" means that there is nothing in the graph itself.

**Adding Lines: The abline() Function:**

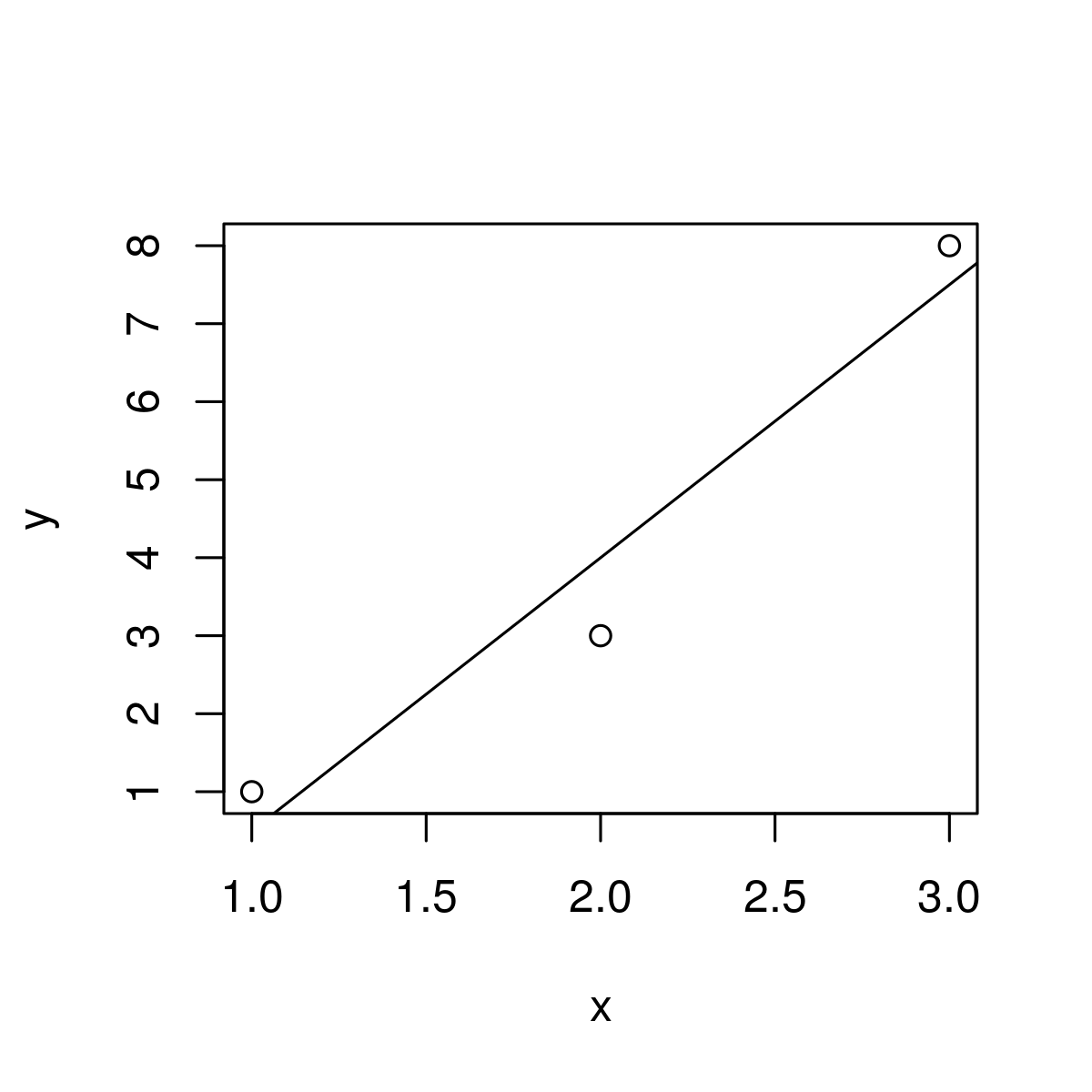
x <- c(1,2,3)

y <- c(1,3,8)

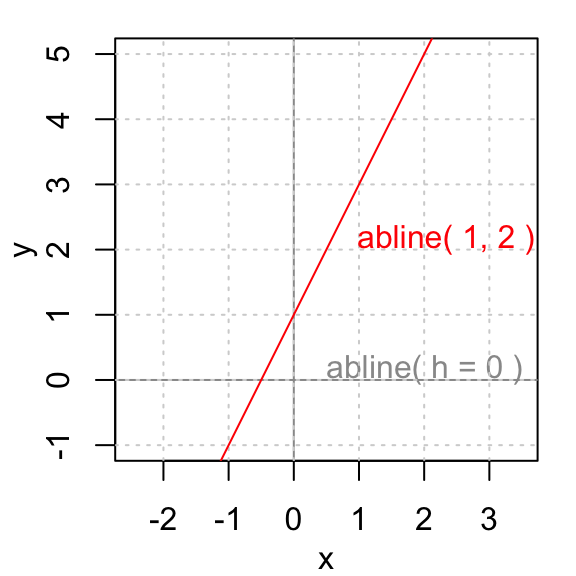
plot(x,y)

lmout <- lm(y ~ x)

abline(lmout)

Using abline()****

How to add one or more **straight lines** to a **graph** using **R statistical software**. The R function **abline()** can be used to add **vertical**, **horizontal** or **regression** **lines** to a graph.



A simplified format of the **abline()** function is :

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

* **a, b** : single values specifying the **intercept** and the **slope** of the line
* **h** : the **y-value(s)** for horizontal line(s)
* **v** : the **x-value(s)** for vertical line(s)

**Add a vertical line**

The simplified format is :

abline(v = y)

It draws a vertical line on the current plot at the specified ‘y’ coordinates.

# first example : Add one line

plot(cars)

abline(v=15, col="blue")

# second example : add 2 lines

# change line colors, sizes and types

plot(cars)

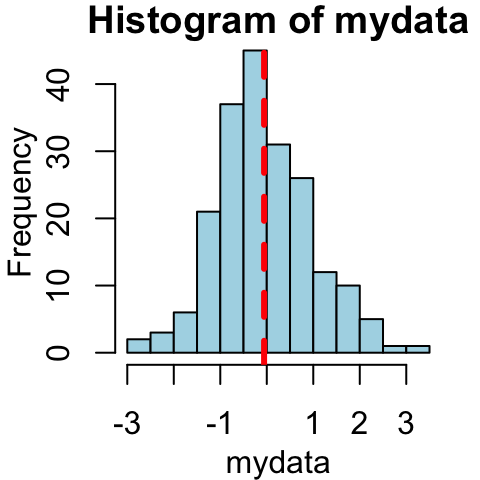
abline(v=c(15,20), col=c("blue", "red"), lty=c(1,2), lwd=c(1, 3))

# third example

set.seed(1234); mydata<-rnorm(200)

hist(mydata, col="lightblue")

abline(v = mean(mydata), col="red", lwd=3, lty=2)



**Add a horizontal line**

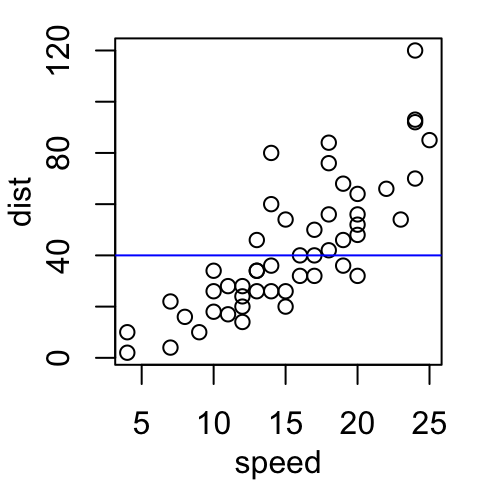
The simplified format is :

abline(h = x)

It draws an horizontal line on the current plot at the specified ‘x’ coordinates.

plot(cars)

abline(h=40, col="blue")



**Add regression line**

**lm()** function is used to fit linear models.

par(mgp=c(2,1,0), mar=c(3,3,1,1))

# Fit regression line

**require**(stats)

reg<-lm(dist ~ speed, data = cars)

coeff=coefficients(reg)

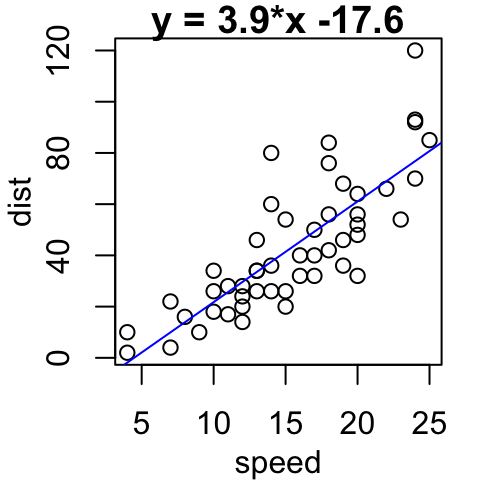
# equation of the line :

eq = paste0("y = ", round(coeff[2],1), "\*x ", round(coeff[1],1))

# plot

plot(cars, main=eq)

abline(reg, col="blue")



**Extended Example: Two Density Estimates on the Same Graph:**

d=data.frame(

v1=c(2.0,3.3,4.0,2.3,2.3,3.3),

v2=c(0.0,1.0,2.0,3.0,4.0,3.7),

v3=c(4.0,3.7,4.0,3.3,3.3,4.0))

d

e1=d

e1

d1 = density(e1$v1,from=0.0,to=10.0)

d1

d2 = density(e1$v2,from=0.0,to=10.0)

d2

plot(d1,main="",xlab="")

lines(d2)

v1 v2 v3

1 2.0 0.0 4.0

2 3.3 1.0 3.7

3 4.0 2.0 4.0

4 2.3 3.0 3.3

5 2.3 4.0 3.3

6 3.3 3.7 4.0

v1 v2 v3

1 2.0 0.0 4.0

2 3.3 1.0 3.7

3 4.0 2.0 4.0

4 2.3 3.0 3.3

5 2.3 4.0 3.3

6 3.3 3.7 4.0

Call:

density.default(x = e1$v1, from = 0, to = 10)

Data: e1$v1 (6 obs.); Bandwidth 'bw' = 0.4694

x y

Min. : 0.0 Min. :0.0000000

1st Qu.: 2.5 1st Qu.:0.0000000

Median : 5.0 Median :0.0009284

Mean : 5.0 Mean :0.0999022

3rd Qu.: 7.5 3rd Qu.:0.1977474

Max. :10.0 Max. :0.4285535

Call:

density.default(x = e1$v2, from = 0, to = 10)

Data: e1$v2 (6 obs.); Bandwidth 'bw' = 0.9908

x y

Min. : 0.0 Min. :0.0000000

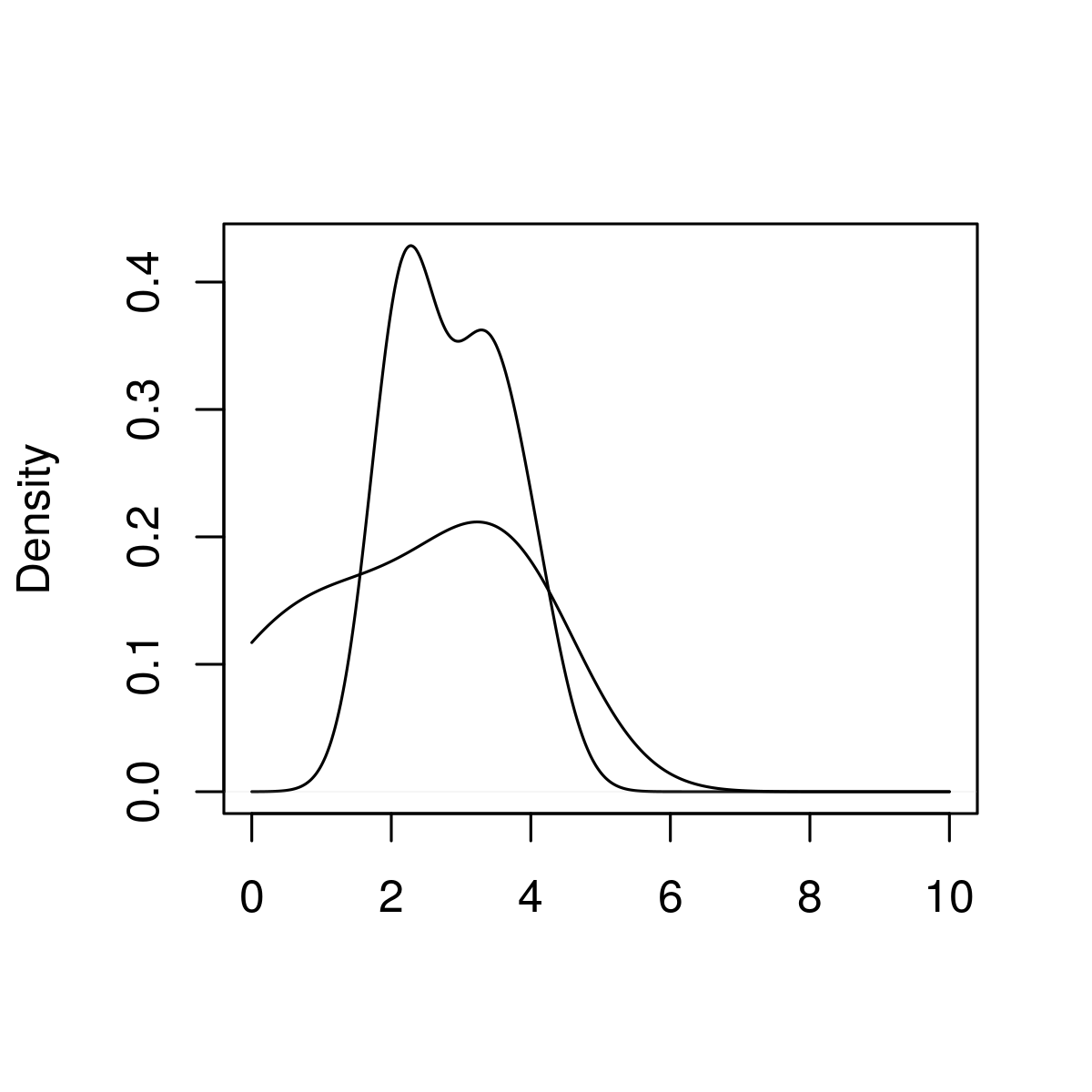
1st Qu.: 2.5 1st Qu.:0.0001788

Median : 5.0 Median :0.0783001

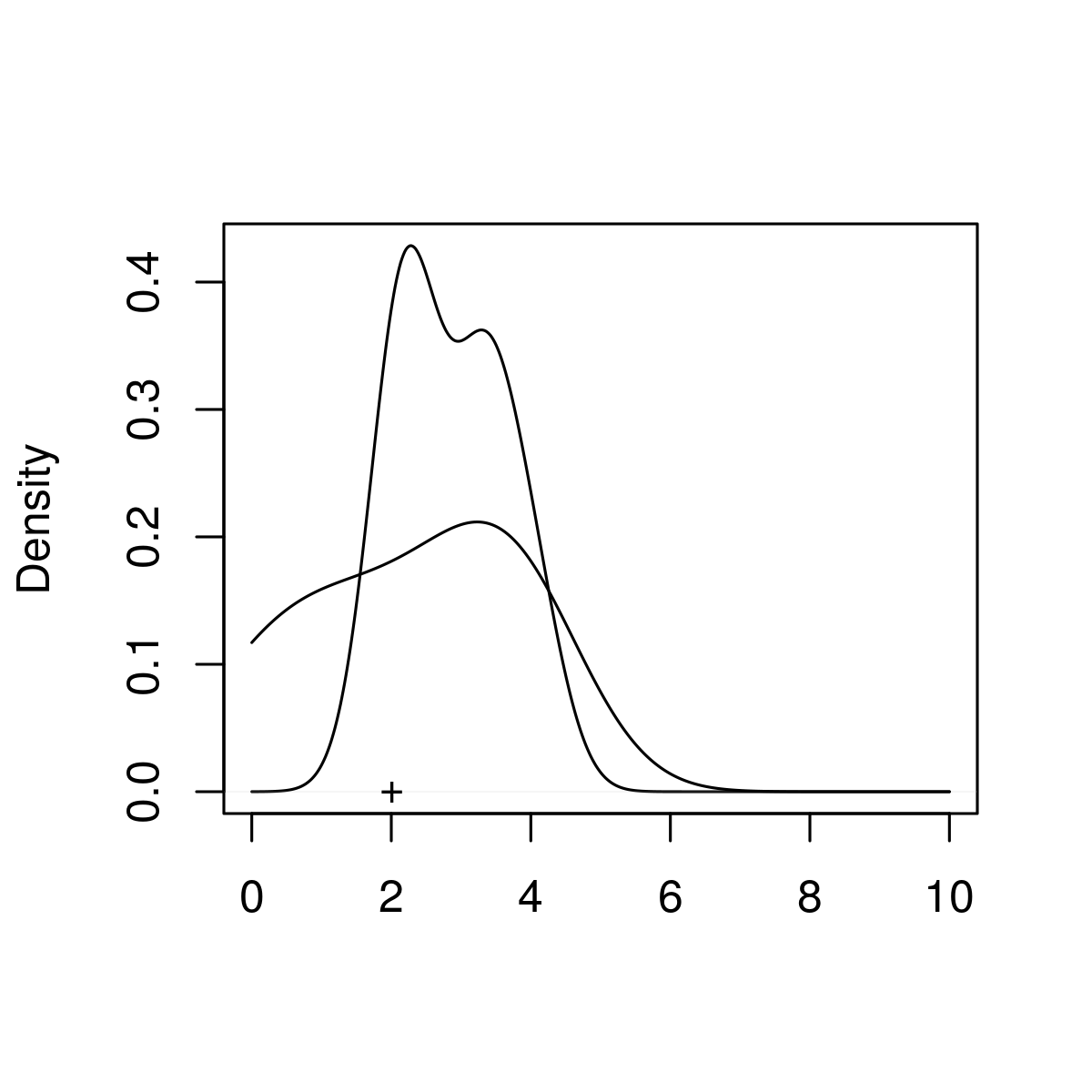
Mean : 5.0 Mean :0.0886947

3rd Qu.: 7.5 3rd Qu.:0.1716352

Max. :10.0 Max. :0.2117307

****

points(testscores$Exam1,testscores$Exam3,pch="+")

****

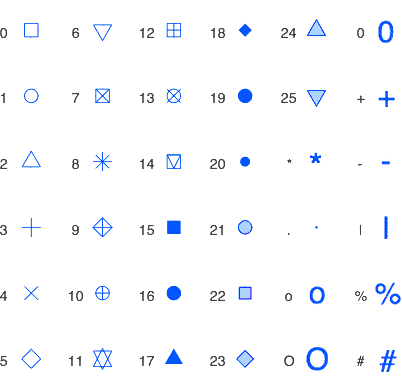
**Adding Functions in R**

1. The points() Function
2. The legend() Function
3. The text() Function
4. The locator() Function

**1. The points() Function**

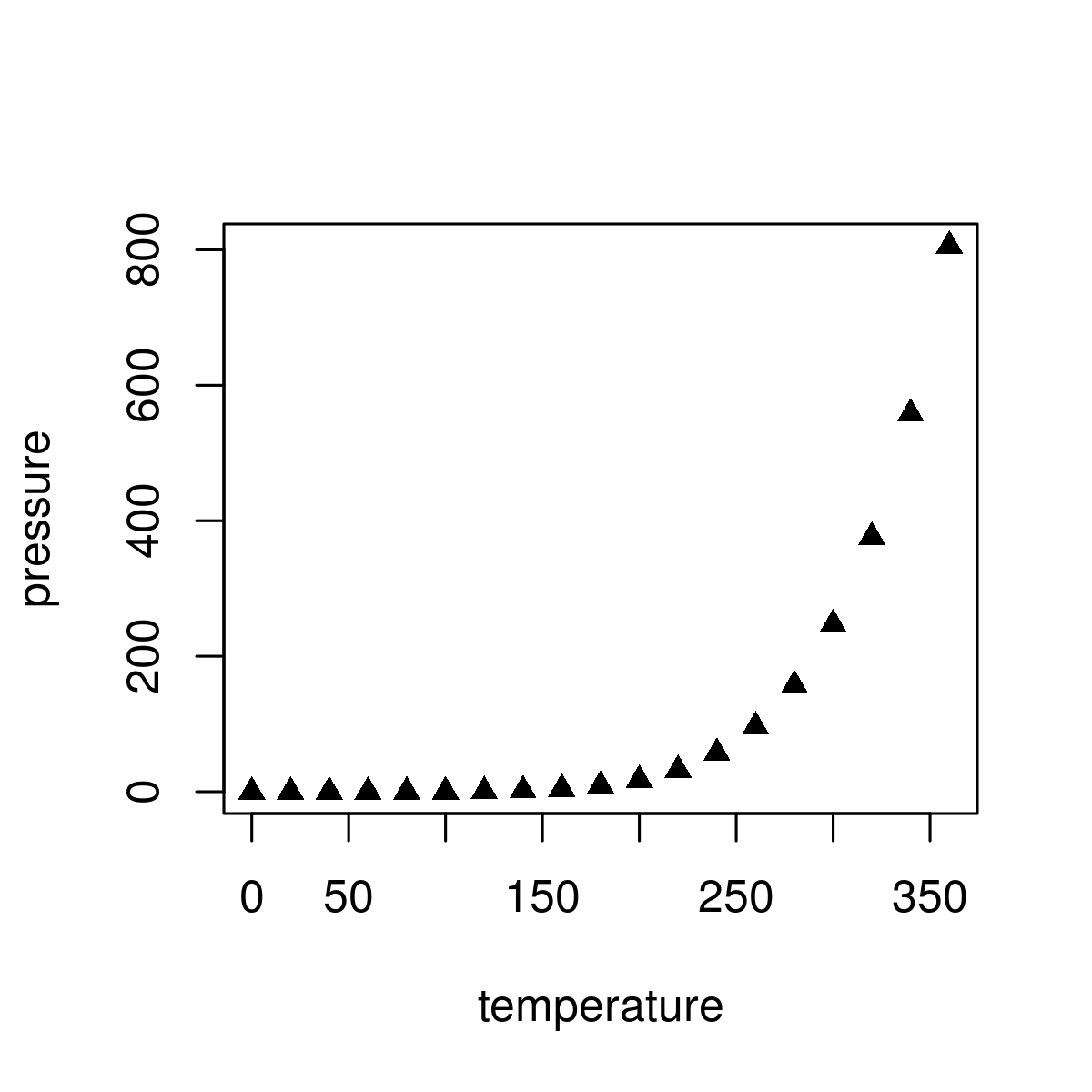
## **Change the Shape and Size of the Points**

You can use the pch (plotting character) argument to specify symbols to use when plotting points.Here’s a list of symbols you can use.

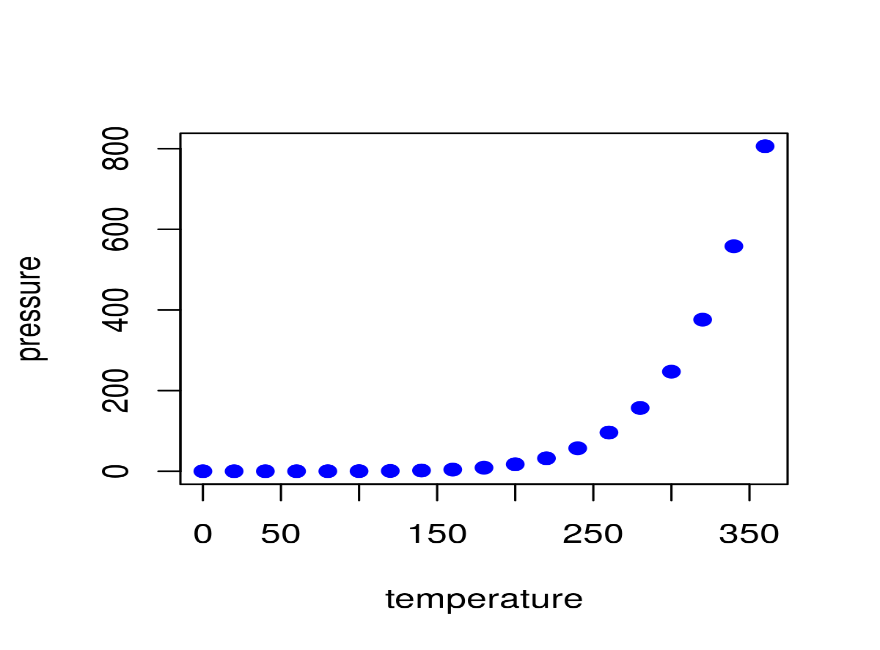


**Ex:**

plot(pressure, pch=17)



plot(pressure, pch=21, col="blue", bg="blue")



## **Different Plot Types**

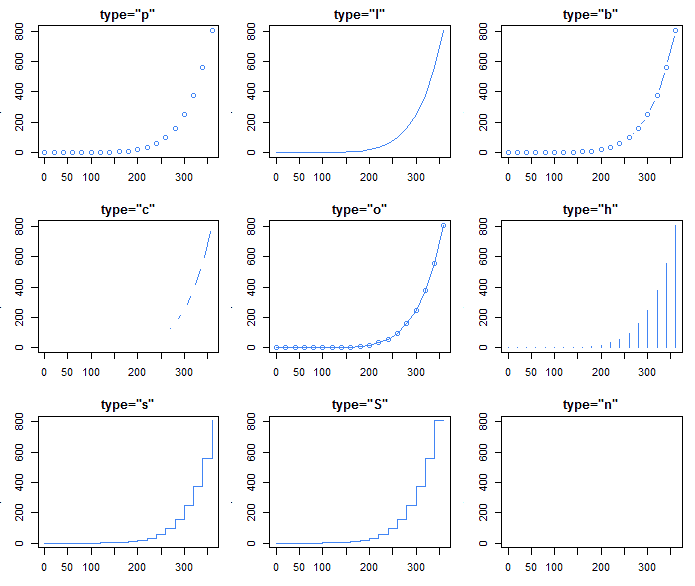
You can change the type of plot that gets drawn by using the type argument.

Here’s a list of all the different types that you can use.

|  |  |
| --- | --- |
| Value | Description |
| “p” | Points |
| “l” | Lines |
| “b” | Both points and lines |
| “c” | The lines part alone of “b” |
| “o” | Both points and lines “overplotted” |
| “h” | Histogram like (or high‐density) vertical lines |
| “s” | Step plot (horizontal first) |
| “S” | Step plot (vertical first) |
| “n” | No plotting |

For example, to create a plot with lines between data points, use type="l"; to draw both lines and points, use type="b".

A series of graphics showing different types is shown below.



## **Adding Titles and Axis Labels**

You can add your own title and axis labels easily by specifying following arguments.

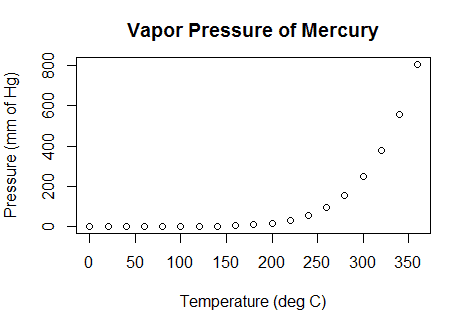
|  |  |
| --- | --- |
| Argument | Description |
| main | Main plot title |
| xlab | x-axis label |
| ylab | y-axis label |

plot(pressure,

main = "Vapor Pressure of Mercury",

xlab = "Temperature (deg C)",

ylab = "Pressure (mm of Hg)")



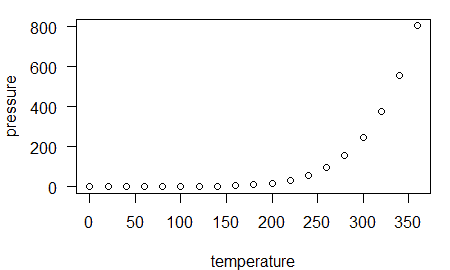
## **The Axes Label Style**

By specifying the las (label style) argument, you can change the axes label style. This changes the orientation angle of the labels.

|  |  |
| --- | --- |
| Value | Description |
| 0 | The default, parallel to the axis |
| 1 | Always horizontal |
| 2 | Perpendicular to the axis |
| 3 | Always vertical |

For example, to change the axis style to have all the axes text horizontal, use las=1

plot(pressure, las = 1)



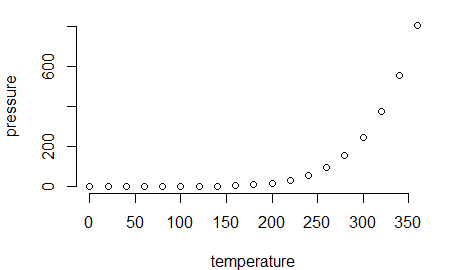
## **The Box Type**

Specify the bty (box type) argument to change the type of box round the plot area.

|  |  |
| --- | --- |
| Value | Description |
| “o” | (default) Draws a complete rectangle around the plot. |
| “n” | Draws nothing around the plot. |
| “l”, “7”, “c”, “u”, or “]” | Draws a shape around the plot area. |

# Remove the box round the plot

plot(pressure, bty="n")

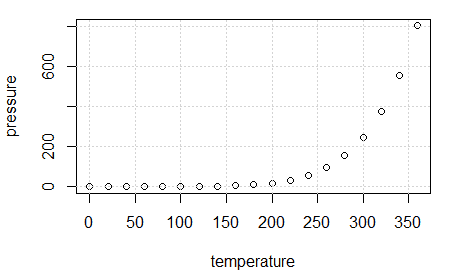


## **Add a Grid**

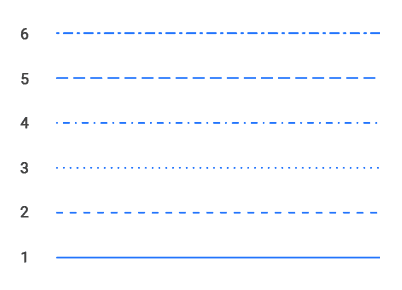
The plot() function does not automatically draw a grid. However, it is helpful to the viewer for some plots. Call the grid() function to draw the grid once you call the plot().

plot(pressure)

grid()



**If you want to specify interms of line so we can use the line number as follows:**



## **Label Data Points**

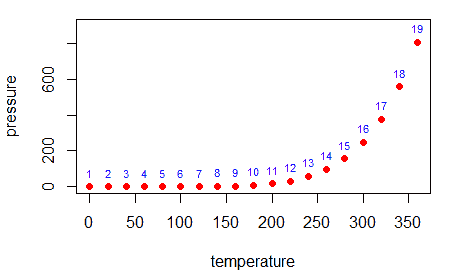
Use the text() function to add text labels at any position on the plot.

The position of the text is specified by the pos argument. Values of 1, 2, 3 and 4, respectively places the text below, to the left of, above and to the right of the specified coordinates.

# Add text labels above the coordinates

plot(pressure, pch=19, col="red")

text(pressure, labels=pressure$pressure, cex=0.7, pos=3, col="blue")



**General Example of points:**

before <- **c**(2.1, 3.5, 1.8, 4.2, 2.4, 3.9, 2.1, 4.4)

after <- **c**(7.5, 5.1, 6.9, 3.6, 7.5, 5.2, 6.1, 7.3)

*# Create plotting space and before scores*

**plot**(x = **rep**(1, **length**(before)),

y = before,

xlim = **c**(.5, 2.5),

ylim = **c**(0, 11),

ylab = "Score",

xlab = "Time",

main = "Using segments() to connect points",

xaxt = "n")

*# Add after scores*

**points**(x = **rep**(2, **length**(after)), y = after)

*# Add connections with segments()*

**segments**(x0 = **rep**(1, **length**(before)),

y0 = before,

x1 = **rep**(2, **length**(after)),

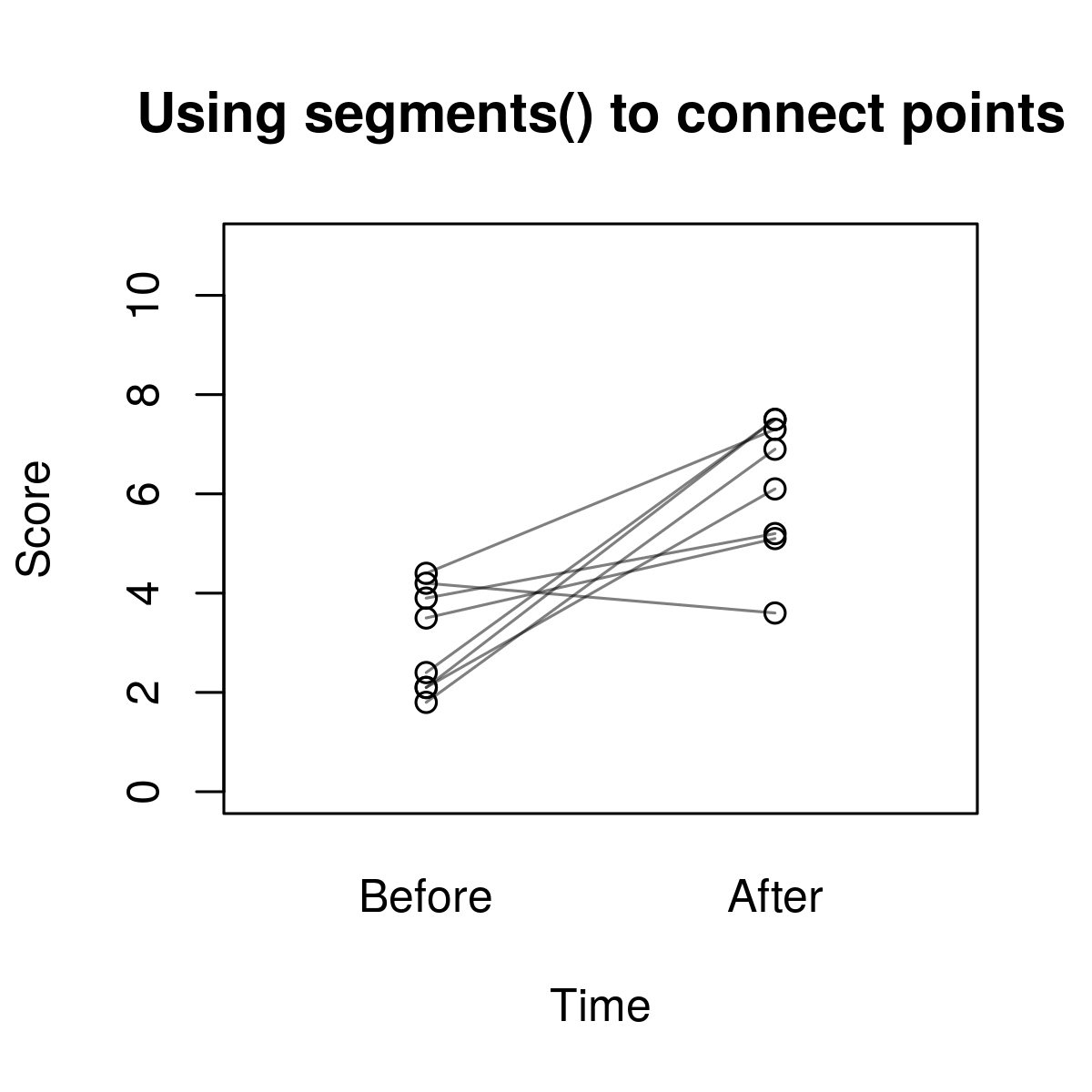
y1 = after,

col = **gray**(0, .5))

*# Add labels*

**mtext**(text = **c**("Before", "After"),

side = 1, at = **c**(1, 2), line = 1)



**2. R legend function**

To add **legends** to plots in **R**, the **R legend()** function can be used. A simplified format of the function is :

legend(x, y=NULL, legend, fill, col, bg)

* **x and y** : the x and y co-ordinates to be used to position the legend
* **legend** : the text of the legend
* **fill** : colors to use for filling the boxes beside the legend text
* **col** : colors of lines and points beside the legend text
* **bg** : the background color for the legend box.

x<-1:10

y1=x\*x

y2=2\*y1

plot(x, y1, type="b", pch=19, col="red", xlab="x", ylab="y")

lines(x, y2, pch=18, col="blue", type="b", lty=2)

legend(1, 95, legend=c("Line 1", "Line 2"), col=c("red", "blue"), lty=1:2, cex=0.8)

# C:\Users\pkumarm6\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\60657DF1.tmp

# Title, text font and background color of the legend box

The arguments below can be used :

* **title**: The title of the legend
* **text.font**: an integer specifying the font style of the legend text; possible values are :
  + **1**: normal
  + **2**: bold
  + **3**: italic
  + **4**: bold and italic
* **bg**: background color of the legend box

x<-1:10

y1=x\*x

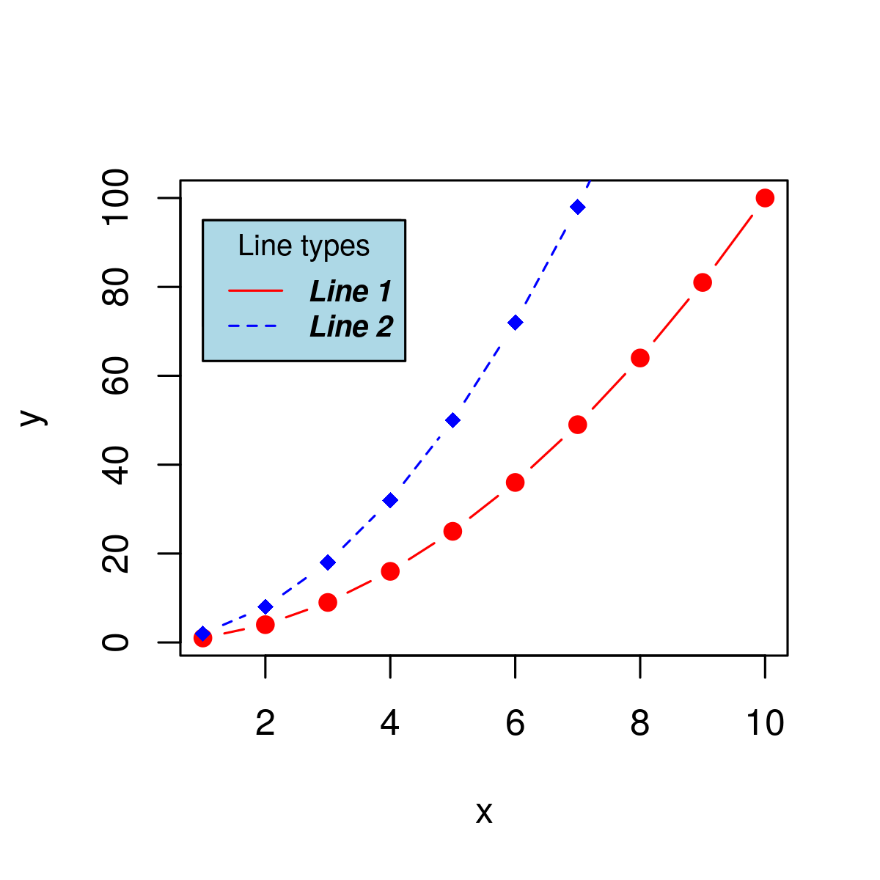
y2=2\*y1

plot(x, y1, type="b", pch=19, col="red", xlab="x", ylab="y")

lines(x, y2, pch=18, col="blue", type="b", lty=2)

legend(1, 95, legend=c("Line 1", "Line 2"), col=c("red", "blue"), lty=1:2, cex=0.8)

legend(1, 95, legend=c("Line 1", "Line 2"), col=c("red", "blue"), lty=1:2, cex=0.8, title="Line types", text.font=4, bg='lightblue')



# Border of the legend box

The arguments **box.lty, box.lwd and box.col** can be used to modify the line type, width and color for the legend box border, respectively.

# x<-1:10

# y1=x\*x

# y2=2\*y1

# plot(x, y1, type="b", pch=19, col="red", xlab="x", ylab="y")

# lines(x, y2, pch=18, col="blue", type="b", lty=2)

# legend(1, 95, legend=c("Line 1", "Line 2"),

# col=c("red", "blue"), lty=1:2, cex=0.8,

# box.lty=0)

# # Change the border

# legend(1, 95, legend=c("Line 1", "Line 2"),

# col=c("red", "blue"), lty=1:2, cex=0.8,

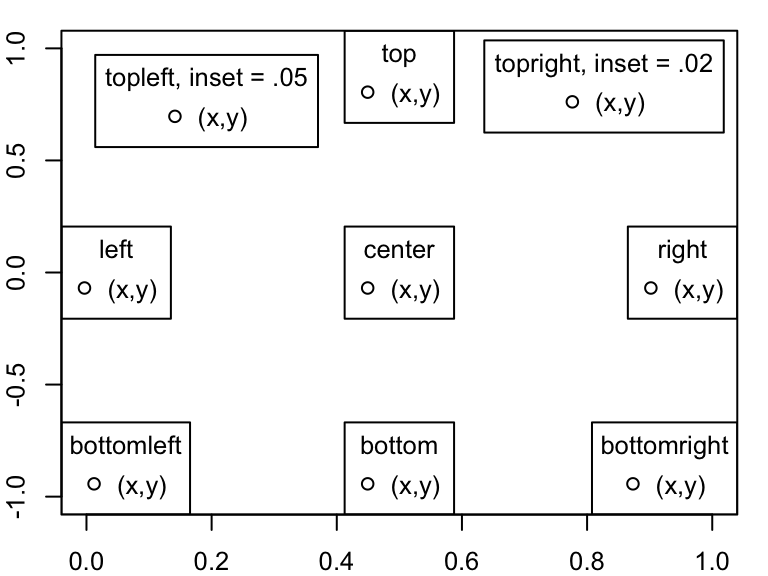
# box.lty=2, box.lwd=2, box.col="green")

# R legend : tutorial on how add legends to plots using R software R legend : tutorial on how add legends to plots using R software

# Specify legend position by keywords

The position of the legend can be specified also using the following keywords : "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center".

The effect of using each of these keywords are shown in the figure below :



## Example 1: line plot

x<-1:10

y1=x\*x

y2=2\*y1

plot(x, y1, type="b", pch=19, col="red", xlab="x", ylab="y")

lines(x, y2, pch=18, col="blue", type="b", lty=2)

legend(1, 95, legend=c("Line 1", "Line 2"),

col=c("red", "blue"), lty=1:2, cex=0.8,

box.lty=0)

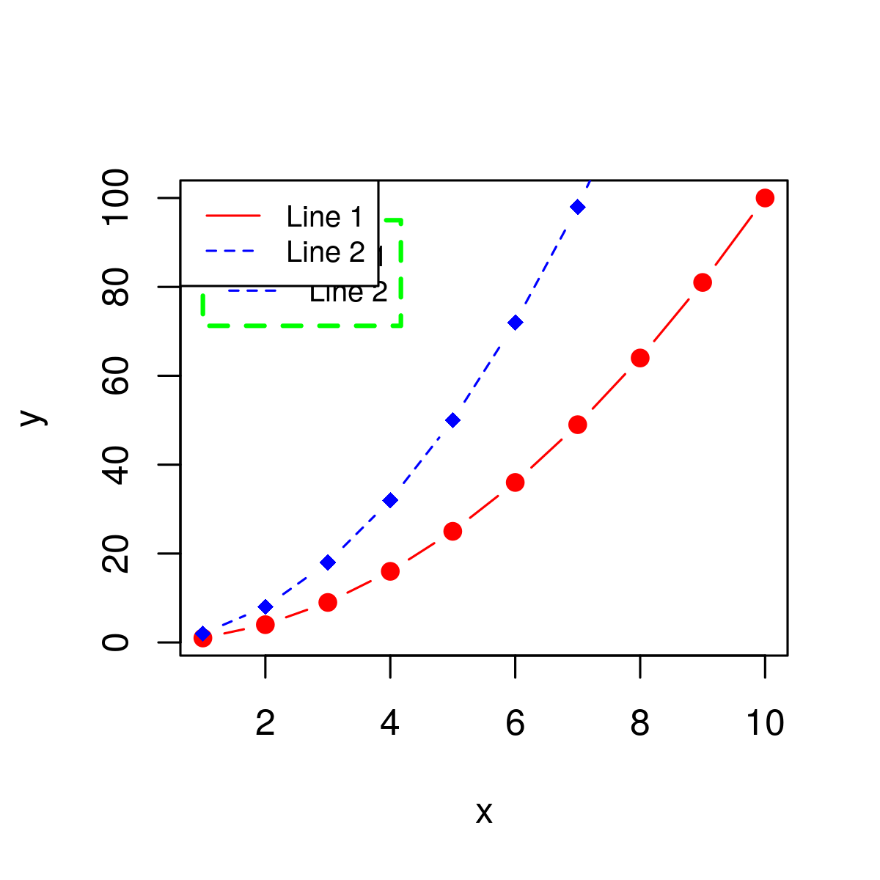
# Change the border

legend(1, 95, legend=c("Line 1", "Line 2"),

col=c("red", "blue"), lty=1:2, cex=0.8,

box.lty=2, box.lwd=2, box.col="green")

legend("topleft", legend=c("Line 1", "Line 2"), col=c("red", "blue"), lty=1:2, cex=0.8)



**3. The text() Function:**

| **Argument** | **Outcome** |
| --- | --- |
| x, y | Coordinates of the labels |
| labels | Labels to be plotted |
| cex | Size of the labels |
| adj | Horizontal text adjustment. adj = 0 is left justified, adj = .5 is centered, and adj = 1 is right-justified |
| pos | Position of the labels relative to the coordinates. pos = 1, puts the label below the coordinates, while 2, 3, and 4 put it to the left, top and right of the coordinates respectively |

With text(), you can add text to a plot. You can use text() to highlight specific points of interest in the plot, or to add information (like a third variable) for every point in a plot. For example, the following code adds the three words “Put”, “Text”, and “Here” at the coordinates (1, 9), (5, 5), and (9, 1) respectively. See Figure [11.11](https://bookdown.org/ndphillips/YaRrr/low-level-plotting-functions.html#fig:puttexthere) for the plot:

plot(1,

xlim = c(0, 10),

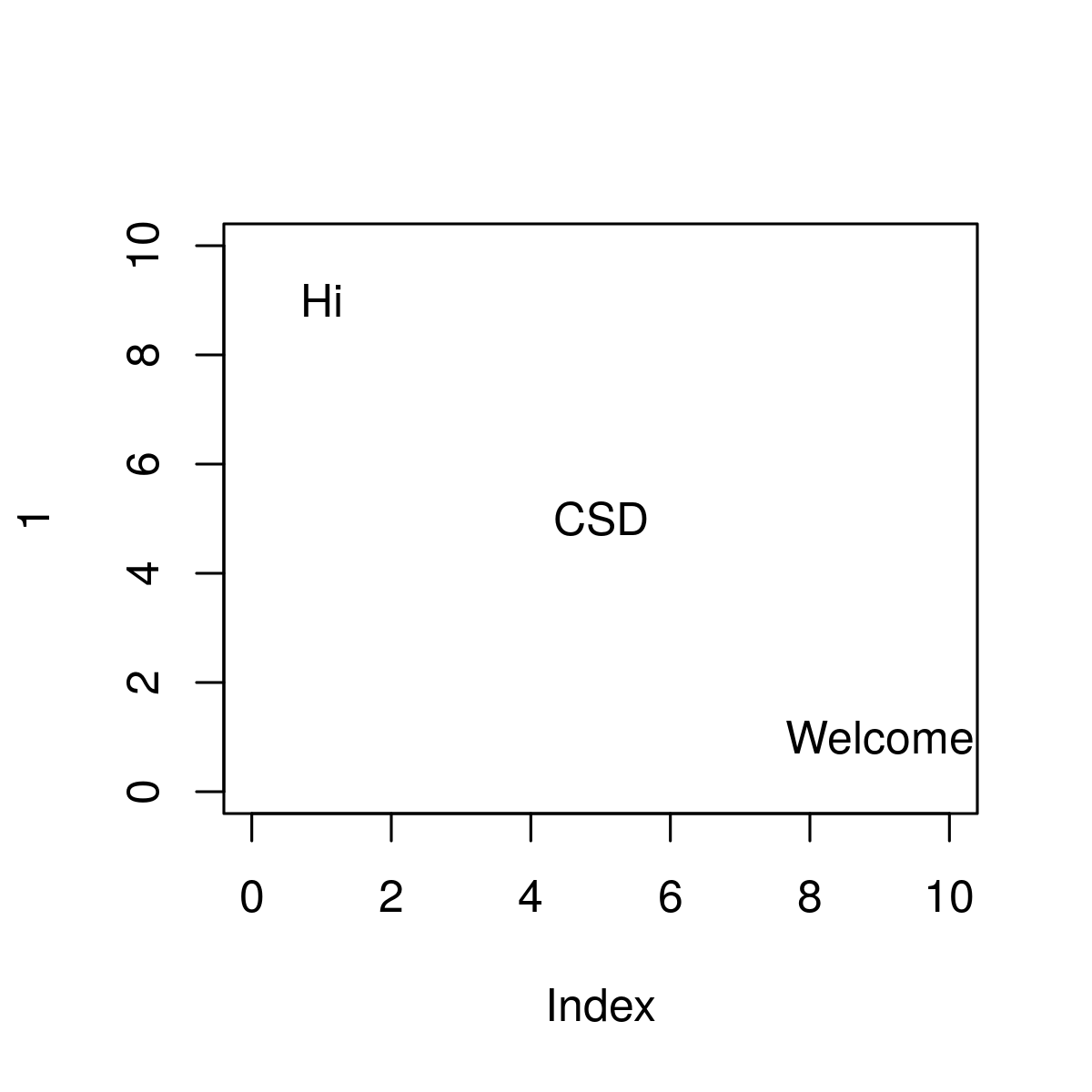
ylim = c(0, 10),

type = "n")

text(x = c(1, 5, 9),

y = c(9, 5, 1),

labels = c("Hi", "CSD", "Welcome"))

****

plot(1,

type = "n",

main = "The \\n tag",

xlab = "", ylab = "")

# Text withoutbreaks

text(x = 1, y = 1.3, labels = "CSD\\n", font = 2)

text(x = 1, y = 1.2,

labels = "Hi Hello Welcome to CSD for R Programming",

font = 3) # italic font

abline(h = 1, lty = 2)

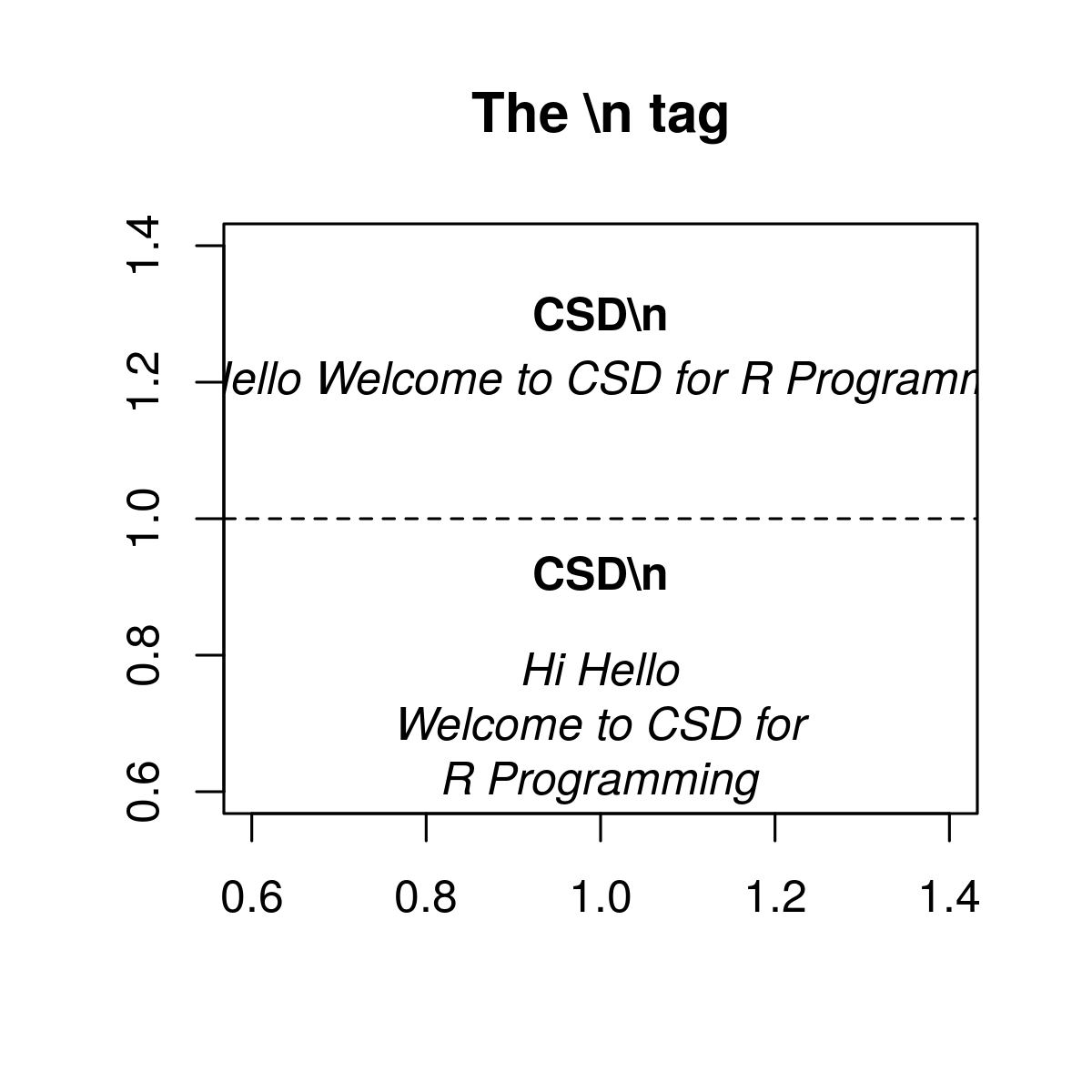
# Text with breaks

text(x = 1, y = .92, labels = "CSD\\n", font = 2)

text(x = 1, y = .7,

labels = "Hi Hello\nWelcome to CSD for\nR Programming",

font = 3) # italic font

****

plot(1, xlim = c(1, 100), ylim = c(1, 100),

type = "n", xaxt = "n", yaxt = "n",

ylab = "", xlab = "", main = "Adding simple figures to a plot")

text(25, 95, labels = "rect()")

rect(xleft = 10, ybottom = 70,

xright = 40, ytop = 90, lwd = 2, col = "coral")

text(25, 60, labels = "polygon()")

polygon(x = runif(6, 15, 35),

y = runif(6, 40, 55),

col = "skyblue")

text(25, 30, labels = "segments()")

segments(x0 = runif(5, 10, 40),

y0 = runif(5, 5, 25),

x1 = runif(5, 10, 40),

y1 = runif(5, 5, 25),

lwd = 2)

text(75, 95, labels = "symbols(circles)")

symbols(x = runif(3, 60, 90),

y = runif(3, 60, 70),

circles = c(1, .1, .3),

add = TRUE, bg = gray(.5, .1))

text(75, 30, labels = "arrows()")

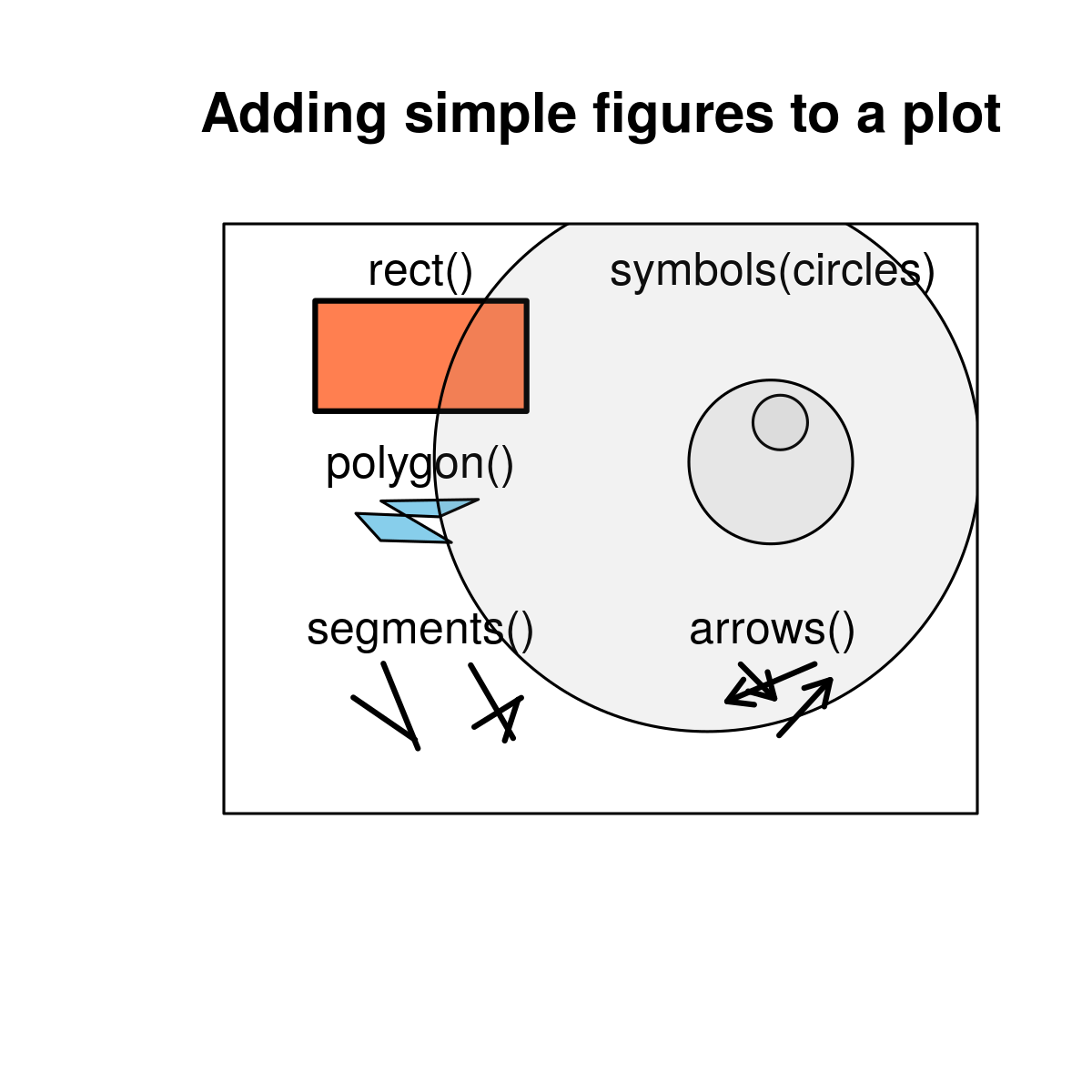
arrows(x0 = runif(3, 60, 90),

y0 = runif(3, 10, 25),

x1 = runif(3, 60, 90),

y1 = runif(3, 10, 25),

length = .1, lwd = 2)

****

1. **The locator() Function:**

The function does a scaling of the x and y axes in order to give them equal weighting and remove the influence of differing units or ranges. The function then **calculates the Euclidean distance between the selected locations** (using the locator () function) and the x, y coordinates of the plotted data points.

plot(runif(100))

legends\_coord <- locator(1) print(legends\_coord)

legend(x= legends\_coord[1], y= legends\_coord[2], legend= "First Legend")

plot(runif(100))

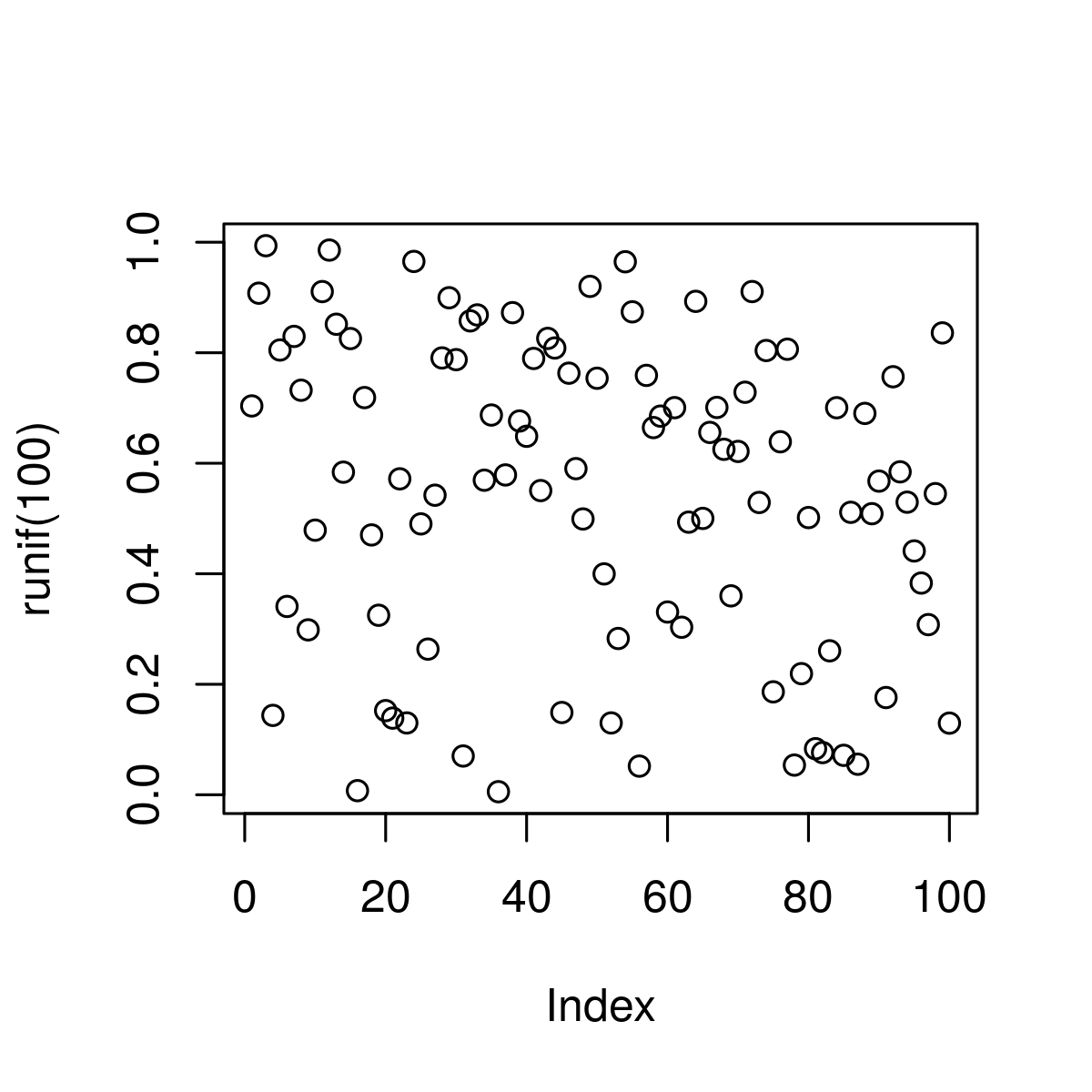
legends\_coord <- locator(3)

print(legends\_coord)

legend(x= legends\_coord$x[1], y= legends\_coord$y[1], legend= "some text", bty="n")

legend(x= legends\_coord$x[2], y= legends\_coord$y[2], legend= "more text", bty="n")

legend(x= legends\_coord$x[3], y= legends\_coord$y[3], legend= "even more text", bty="n")

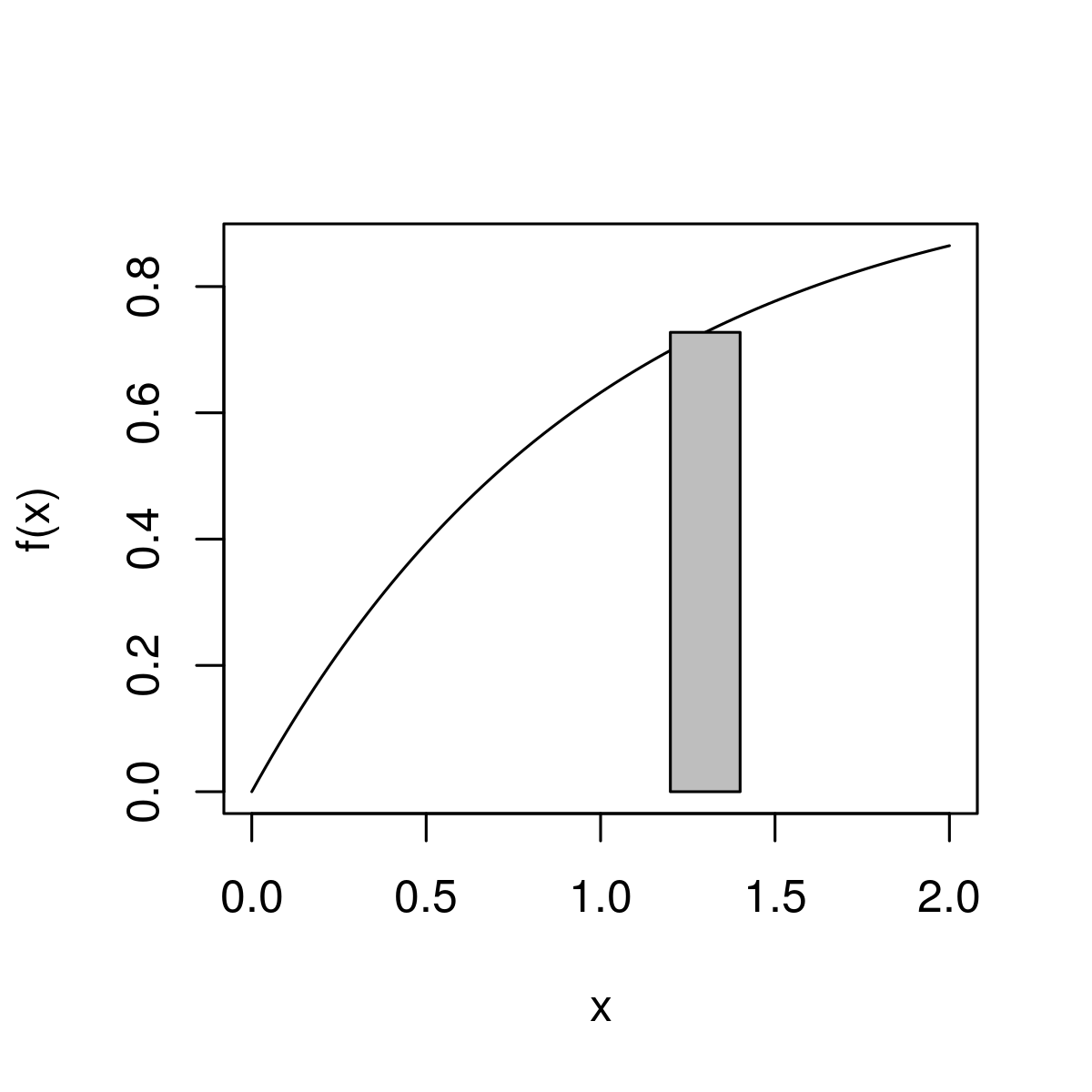


1. **The polygon() Function:**

f <- function(x) return(1-exp(-x))

curve(f,0,2)

polygon(c(1.2,1.4,1.4,1.2),c(0,0,f(1.3),f(1.3)),col="gray")



f <- function(x) return(1-exp(-x))

curve(f,0,2)

polygon(c(1.2,1.4,1.4,1.2),c(0,0,f(1.3),f(1.3)),col="gray")

polygon(c(1.2,1.4,1.4,1.2),c(0,0,f(1.3),f(1.3)),density=10)

