########################## Working with anonymous functions

(function(x){3\*x+2})(1)

(function(x){3\*x+2})(c(-10:10))

########################## A function call with no arguments

fun1 <- function()

{

print( "This is a function without arguments")

cat("2 \* 4 is ",2\*4)

cat(2\*4,"is the result")

print("hello")

print("world")

}

fun1()

######################## A function with 2 arguments but no defaults

fun2 <- function(a,b){

print(a+b)

print(a\*b)

}

#no arguments passed

fun2()

#less no of arguments passed

fun2(1)

#correct no of arguments passed

fun2(1,2)

#more arguments passed

fun2(1,2,)

####################### Defining functions wuth defaults

fun3 <- function(a,b=2){

print(a+b)

print(a\*b)

}

#no arguments passed

fun3()

#less no of arguments passed

fun3(1)

fun3(,1)

fun3(,a=1)

fun3(a=6)

#correct no of arguments passed

fun3(1,2)

fun3(a=12,b=5)

fun3(b=10,a=15)

#more arguments passed

fun3(1,2,)

########################## Ignoring the excessive arguments

fun4 <- function(a=10,b=15,...){

print(a+b)

print(a\*b)

}

#no arguments passed

fun4()

#less no of arguments passed

fun4(1)

fun4(,1)

fun4(,a=1)

fun4(a=6)

#correct no of arguments passed

fun4(1,2)

fun4(a=12,b=5)

fun4(b=10,a=15)

#more arguments passed

fun4(1,2,)

fun4(1,2,3)

##################### Local Vs Global variables

# addab is a local variable and is not available outside the function

fun5 <- function(a=10,b=15,...){

addab<- a+b

print(addab)

}

fun5()

print(addab)

#addab is created in the Global environment

fun5 <- function(a=10,b=15,...){

addab<<- a+b

print(addab)

}

fun5()

print(addab)

################ Passing a function as a argument

intvec <- c(1:10)

mean(intvec)

mean(intvec,na.rm=T)

intvec[10] <- NA

intvec

mean(intvec)

mean(intvec,na.rm=T)

df <- data.frame(

v1 = c(1:100),

v2 = c(101:200),

v3 = c(201:300)

)

head(df)

summary(df)

fun6 <- function(df,fun=mean,...){

for(i in 1: length(df)){

print(fun(df[[i]]))

}

}

fun6(df)

fun6(df,na.rm=T)

df$v1[1] <- NA

summary(df)

fun6(df)

fun6(df,na.rm=T)

fun6(df,123)

fun6(df,mean,123)

fun6(df,T)

# using the optional arguments in the function

fun6 <- function(df,fun=mean,...){

for(i in 1: length(df)){

print(fun(df[[i]],...))

}

}

fun6(df)

fun6(df,na.rm=T)

mean(c(2:100))

# Can you figure out why the answer with trim and without trim are the same

fun6(df,na.rm=T,trim=.1)

# example of inbuilt functions that eliminate the need for functions

sapply(df,mean,na.rm=T)

lapply(df,mean,na.rm=T)