lecture five

Yi (Chris) Chen

October 6, 2017

## lecture five

#### Writing Functions In R

* the priciple of writing function:

1. be fairly short
2. focus and specific in what it dose
3. designed so that it can be used in many ways

* basic structure: function\_name <- function(arg1,arg2,...){ statements return(object) }

##### example1 of function

# A Function to Check for Significance at ?? = 0:05  
significant <- function(x){  
 if(x <= 0.05){return(TRUE)}  
 else{return(FALSE)}  
}  
significant(1)

## [1] FALSE

significant(0.001)

## [1] TRUE

##### example2 of funciton

# A `Robust' Loss Function (for Outlier-Resistant Regression)  
  
res\_loss <- function(x) {  
 loss\_vec <- ifelse(x^2 > 1, 2\*abs(x) - 1, x^2)  
 return(loss\_vec)  
}  
  
vec <- c(-0.5, 0.9, -3, 4)  
res\_loss(vec)

## [1] 0.25 0.81 5.00 7.00

##### task of function

Write a function called FiveTimesSum that takes as input a vector of numerical values and returns 5 times the sum of those values. Test it on the vector 1:3. Your output should be 30.

FiveTimesSum\_1 <- function(x){  
 result <- sum(5\*(x))  
 return(result)  
}  
vec <- 1:3  
FiveTimesSum\_1(vec)

## [1] 30

# this can also be solved like this  
FiveTimesSum\_2 <- function(x){  
 for(i in 1:length(x)){  
 x[i] = x[i] \*5  
 }  
 result <- sum(x)  
 return(result)  
}  
vec <- 1:3  
FiveTimesSum\_2(vec)

## [1] 30

##### Named and Default Arguments

# Inputs: A vector of numbers (x)  
# crossover location (c>0)  
# Outputs: A loss vector with x^2 for small elements(which is smaller than c),  
# and 2|x|-c for large ones(which is bigger than c)  
res\_loss2 <- function(x, c = 1) {  
 loss\_vec <- ifelse(x^2 > c, 2\*c\*abs(x) - c, x^2)  
 return(loss\_vec)  
}  
  
identical(res\_loss(vec), res\_loss2(vec, c = 1))

## [1] TRUE

identical(res\_loss(vec), res\_loss2(vec, c = 2))

## [1] FALSE

identical(res\_loss2(vec, c = 1), res\_loss2(vec))

## [1] TRUE

identical(res\_loss2(x = vec, c = 2),res\_loss2(c = 2, x = vec))

## [1] TRUE

vec <- c(-0.5, 0.9, -3, 4)  
res\_loss2(vec, c = c(1,1,1,5)) # they input the value of c respectively. Actually c only make sense when it is a single positive number

## [1] 0.25 0.81 5.00 35.00

res\_loss2(vec, c = -1) # even if c can not be a negative number but the function could still work

## [1] 0.0 -0.8 -5.0 -7.0

* if you want to add some check in your function about the range of c, you can do in this way

res\_loss3<- function(x,c=1) {  
 # Scale should be a single positive number  
 stopifnot(length(c)==1,c>0) # stopifnot if the situdation can not be satisfied.  
 loss\_vec <- ifelse(x^2 > c, 2\*abs(x) - c, x^2)  
 return(loss\_ve2)  
}

#### task

Write a function the will take an input vector and set any value below a threshold to be the value of the threshold. Optionally, the function should instead set values above the threshold to the value of the threshold. Hint: The function should have three arguments, two required the vector and the threshold, and one optional with a default value of "below".

threshold\_function <- function(vec,thresh,dir='below'){  
 stopifnot(dir == 'below' | dir == 'above')  
 if(dir=='below') {  
 vec[vec<thresh] <- thresh  
 }else{  
 vec[vec>thresh] <- thresh  
 }  
 return(vec)  
}  
vec <- 1:10  
threshold\_function(vec,thresh = 5,dir = 'below')

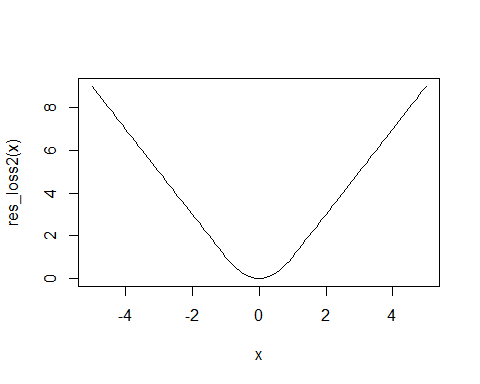
## [1] 5 5 5 5 5 6 7 8 9 10

threshold\_function(vec,thresh = 5,dir = 'above')

## [1] 1 2 3 4 5 5 5 5 5 5

### Use Your Functions in Other Functions

# automatically try the different value in the function  
curve(res\_loss2,from = -5 , to= 5)



#### grobel environment

x <- 2  
y <- function(y){  
 return(x + y)  
}  
y(1) # there is no value of x inside the function so that it will find the value of x in the golbel enviornment

## [1] 3

g <- function(y){  
 x <-10  
 return(x+y)  
}  
g(1) # it has the value of x in the function so it will not use the value in the gobel environment

## [1] 11

g<- function(y){  
 f <- function(y){ # the f function is not in the gobel enivorment  
 return(x+y)  
 }  
 x<-10 # x is not in the gobel environment  
 return(f(y))  
}  
g(1)

## [1] 11

#### everything in R is a function and everything in R is an object

'+'(3,2)

## [1] 5

class(1)

## [1] "numeric"

##### extended example: fitting a model

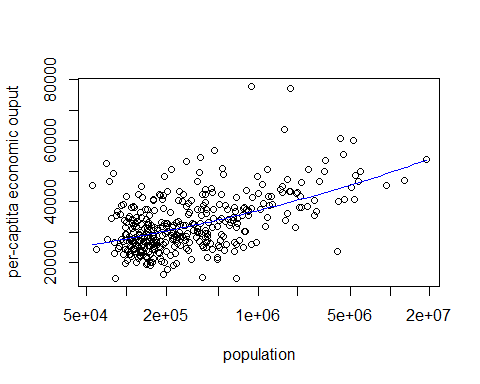
setwd("C:/Users/cheny/Desktop/study/statistical computing and intro to data science/lecture five")  
gmp <- read.table('gmp.txt',as.is = TRUE,header = TRUE) # as.is means we want or not to make the character as factor  
head(gmp)[1:3,]

## city gmp pcgmp  
## 1 Abilene, TX 3.8870e+09 24490  
## 2 Akron, OH 2.2998e+10 32889  
## 3 Albany, GA 3.9550e+09 24269

gmp$pop <- gmp$gmp/gmp$pcgmp  
head(gmp,5)

## city gmp pcgmp pop  
## 1 Abilene, TX 3.8870e+09 24490 158717.8  
## 2 Akron, OH 2.2998e+10 32889 699261.2  
## 3 Albany, GA 3.9550e+09 24269 162965.1  
## 4 Albany-Schenectady-Troy, NY 3.1321e+10 36836 850282.3  
## 5 Albuquerque, NM 3.0727e+10 37657 815970.5

plot(gmp$pop,gmp$pcgmp,log = 'x',xlab='population',ylab = 'per-captita economic ouput')  
  
# beta\_0 =6611,beta\_1=1/8  
curve(6611\*x^{1/8},add = TRUE,col="blue")



##### fitting the function

* strategy:  
  minimize the mean sum of squares of training data
* assume:  
  that it is a convex function: twice differentiable function is always bigger than 0 and there is only one minimum (both local and global)
* algorith:  
  gradient descent search for a minimum of f

1. start with some point x and fix a precision b bigger than 0
2. repeat for n= 1,2,3... that x(n+1)=x(n)- c \* f'(x(n)), where c is the scale of step size
3. terminate when |f'(x(n))| < b

##### first attemp at code

max.iter <- 100 # how long we run the alg  
stop.deriv <- 1/100 # if derivative is samll, stop (b)  
deriv.step <- 1/1000 # that is the samll h which is used to calculate the differente  
step.scale <- 1e-12 # scale of step size (c)  
  
iter <- 0 # iteration counter  
deriv <- Inf  
beta <- 0.15 # start point   
  
while((iter < max.iter) & (deriv > stop.deriv)) {  
 # the function will stop if the iteration is too long or the deriv is samller than the b  
 iter <- iter + 1  
 # calculate the differente  
 mse.1 <- mean((gmp$pcgmp - 6611\*gmp$pop^beta)^2)  
 mse.2 <- mean((gmp$pcgmp - 6611\*gmp$pop^(beta + deriv.step))^2)  
 deriv <- (mse.2 - mse.1)/deriv.step  
 # update the value of beta  
 beta <- beta - step.scale\*deriv  
}  
list(beta = beta, iteration = iter, conv = (iter < max.iter))

## $beta  
## [1] 0.1258166  
##   
## $iteration  
## [1] 58  
##   
## $conv  
## [1] TRUE

##### first fix

problems: 1. not encapsulated 2. inflexiable 3. error-pone 4. hard to fix

est.exp <- function(beta) {  
 max.iter <- 100 # how long we run the alg  
 stop.deriv <- 1/100 # if derivative is samll, stop (b)   
 deriv.step <- 1/1000 # that is the samll h which is used to calculate the differente  
 step.scale <- 1e-12 # scale of step size (c)  
 iter <- 0 # iteration counter  
 deriv <- Inf  
  
 while((iter < max.iter) & (deriv > stop.deriv)) {  
 # the function will stop if the iteration is too long or the deriv is samller than the b  
 iter <- iter + 1  
 # calculate the differente  
 mse.1 <- mean((gmp$pcgmp - 6611\*gmp$pop^beta)^2)  
 mse.2 <- mean((gmp$pcgmp - 6611\*gmp$pop^(beta + deriv.step))^2)  
 deriv <- (mse.2 - mse.1)/deriv.step  
 # update the value of beta  
 beta <- beta - step.scale\*deriv  
 }  
 fit <- list(beta = beta, iteration = iter, conv = (iter < max.iter))  
 return(fit)  
}  
   
est.exp(0.15)

## $beta  
## [1] 0.1258166  
##   
## $iteration  
## [1] 58  
##   
## $conv  
## [1] TRUE

###### second fix

problem: 1. have to return if we want to change defined pararmeters

est.exp <- function(beta, beta\_0 = 6611, max.iter = 100,  
stop.deriv = .01, deriv.step = .001,  
step.scale = 1e-12) {  
  
 iter <- 0 # iteration counter  
 deriv <- Inf  
  
 while((iter < max.iter) & (deriv > stop.deriv)) {  
 # the function will stop if the iteration is too long or the deriv is samller than the b  
 iter <- iter + 1  
 # calculate the differente  
 mse.1 <- mean((gmp$pcgmp - 6611\*gmp$pop^beta)^2)  
 mse.2 <- mean((gmp$pcgmp - 6611\*gmp$pop^(beta + deriv.step))^2)  
 deriv <- (mse.2 - mse.1)/deriv.step  
 # update the value of beta  
 beta <- beta - step.scale\*deriv  
 }  
 fit <- list(beta = beta, iteration = iter, conv = (iter < max.iter))  
 return(fit)  
}  
   
est.exp(0.15)

## $beta  
## [1] 0.1258166  
##   
## $iteration  
## [1] 58  
##   
## $conv  
## [1] TRUE

###### thrid fix

problem: 1. Problem: Don't need to write out the MSE calculations twice in thebody of the function.

est.exp <- function(beta, beta\_0 = 6611, max.iter = 100,  
stop.deriv = .01, deriv.step = .001,  
step.scale = 1e-12) {  
  
 iter <- 0 # iteration counter  
 deriv <- Inf  
  
 ## add a function  
 mse <- function(b) {mean((gmp$pcgmp - beta\_0\*gmp$pop^b)^2)}  
  
   
 while((iter < max.iter) & (deriv > stop.deriv)) {  
 # the function will stop if the iteration is too long or the deriv is samller than the b  
 iter <- iter + 1  
 # calculate the differente  
 deriv <- (mse(beta + deriv.step) - mse(beta))/deriv.step  
 # update the value of beta  
 beta <- beta - step.scale\*deriv  
 }  
 fit <- list(beta = beta, iteration = iter, conv = (iter < max.iter))  
 return(fit)  
}  
   
est.exp(0.15)

## $beta  
## [1] 0.1258166  
##   
## $iteration  
## [1] 58  
##   
## $conv  
## [1] TRUE

###### fourth fix

problem: 1. Problem: Locked into using specific columns of gmp { if we want touse a di erent data set, have to rewrite the function.

est.exp <- function(beta, beta\_0 = 6611,response = gmp$pcgmp,predictor = gmp$pop, max.iter = 100,stop.deriv = .01, deriv.step = .001,step.scale = 1e-12) {  
  
 iter <- 0 # iteration counter  
 deriv <- Inf  
  
 ## add a function  
 mse <- function(b) {mean((response - beta\_0\*predictor^b)^2)}  
  
   
 while((iter < max.iter) & (deriv > stop.deriv)) {  
 # the function will stop if the iteration is too long or the deriv is samller than the b  
 iter <- iter + 1  
 # calculate the differente  
 deriv <- (mse(beta + deriv.step) - mse(beta))/deriv.step  
 # update the value of beta  
 beta <- beta - step.scale\*deriv  
 }  
 fit <- list(beta = beta, iteration = iter, conv = (iter < max.iter))  
 return(fit)  
}  
   
est.exp(0.15)

## $beta  
## [1] 0.1258166  
##   
## $iteration  
## [1] 58  
##   
## $conv  
## [1] TRUE

###### fifth fix

problem: 1. Want to make it easy for humans to read.

est.exp <- function(beta, beta\_0 = 6611, response = gmp$pcgmp,predictor = gmp$pop, max.iter = 100,stop.deriv = .01, deriv.step = .001,step.scale = 1e-12) {  
 iter <- 0  
 deriv <- Inf  
   
 mse <- function(b) {mean((response - beta\_0\*predictor^b)^2)}  
  
 for (i in 1:max.iter) {  
 iter <- iter + 1  
 deriv <- (mse(beta + deriv.step) - mse(beta))/deriv.step  
 beta <- beta - step.scale\*deriv  
 if (abs(deriv) < stop.deriv) {break()}  
}  
 fit <- list(beta = beta, iteration = iter, conv = (iter < max.iter))  
 return(fit)  
}  
est.exp(0.15)

## $beta  
## [1] 0.1258166  
##   
## $iteration  
## [1] 58  
##   
## $conv  
## [1] TRUE

##### classifiation

library(ISLR)

## Warning: package 'ISLR' was built under R version 3.4.2

head(Smarket,3)

## Year Lag1 Lag2 Lag3 Lag4 Lag5 Volume Today Direction  
## 1 2001 0.381 -0.192 -2.624 -1.055 5.010 1.1913 0.959 Up  
## 2 2001 0.959 0.381 -0.192 -2.624 -1.055 1.2965 1.032 Up  
## 3 2001 1.032 0.959 0.381 -0.192 -2.624 1.4112 -0.623 Down

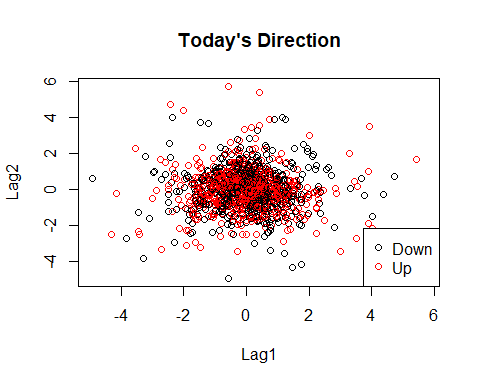
mean(Smarket$Lag1[Smarket$Direction=='Up'])

## [1] -0.03969136

mean(Smarket$Lag1[Smarket$Direction=='Down'])

## [1] 0.05068605

plot(Smarket$Lag1, Smarket$Lag2, col = Smarket$Direction,xlab="Lag1", ylab="Lag2", main="Today's Direction")  
  
# use lag1 and lag2 as example and draw a picture  
legend("bottomright", legend = levels(Smarket$Direction),col=1:length(levels(Smarket$Direction)), pch=1)



KNNclass <- function(L1.new = 2, L2.new = 4.25, k = 5 , L1 = train$Lag1, L2 = train$Lag2, Dir = train$Direction){  
 # k = 5 . new point (2,4.25)  
 dists <- sqrt((L1-L1.new)^2 +(L2 -L2.new)^2)  
 # pick up the nearest k point  
 neighbors <- order(dists)[1:K]  
 neigh.dir <- Dir[neighbors]  
 choice <- names(which.max(table(neigh.dir)))  
 choice  
}