R Programming Problems

Ashik E Elahi

2024-02-19

#### Problem 1:

##### For a = 5, b = 15 and c = 10, find

##### (i) ,

##### (ii) ,

##### (iii) ,

##### (iv)

a = 5; b = 15; c = 10  
# (i)  
(a+b)/c

## [1] 2

# (ii)  
(-b+sqrt(b^2 - 4\*a\*c))/ (2\*a)

## [1] -1

# (iii)  
a\*c + abs(c-b)

## [1] 55

# (iv)  
(2\*a-c)^2

## [1] 0

#### Problem 2

##### (i) Create a data vector in R for 20, 13, 14, 25, 16, 20, 10, 9, 23, 35, 70, 12, 15, 13, 10. Manupulate the data vector by removing 70.

##### (ii) Create a data vector in R which is 1 to 15.

##### (iii) Create a data vector in R which 1 to 24 by 2.

##### (iv) Create a following data matrix in R

Also find (a) row sum, (b) column sum, and (c) the element in (2, 3) position.

# (i) Removing element from data vector  
x <- c(20, 13, 14, 25, 16, 20, 10, 9, 23, 35, 70, 12, 15, 13, 10)  
x <- x[-11]  
print(x)

## [1] 20 13 14 25 16 20 10 9 23 35 12 15 13 10

#(ii) Data vector (1 to 15)  
x2 <- 1:15  
print(x2)

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

#(iii) Data vector (1 to 24 by 2)  
x3 <- seq(1, 24, 2)  
print(x3)

## [1] 1 3 5 7 9 11 13 15 17 19 21 23

#(iv) Matrix   
A <- matrix(1:12, nrow = 3, byrow = T)  
print(A)

## [,1] [,2] [,3] [,4]  
## [1,] 1 2 3 4  
## [2,] 5 6 7 8  
## [3,] 9 10 11 12

# (a) Row Sums   
rowSums(A)

## [1] 10 26 42

# (b) Columns Sums  
colSums(A)

## [1] 15 18 21 24

# (c) Element at (2, 3) position   
A[2, 3]

## [1] 7

#### Problem 3

##### (i) Create the following data matrix in R:

##### (a) Transpose of A.

##### (b) Find the inverse of A, and

##### (c) Eigen values of A

##### (ii) Consider another matrix

##### (a) Find A + B, A - B

##### (b) Find A x B

# (i) First matrix   
A <- matrix(1:12, 3, 4, T)  
A <- A[, -4]  
  
print(A)

## [,1] [,2] [,3]  
## [1,] 1 2 3  
## [2,] 5 6 7  
## [3,] 9 10 11

# (a) Transpose of A   
t(A)

## [,1] [,2] [,3]  
## [1,] 1 5 9  
## [2,] 2 6 10  
## [3,] 3 7 11

# (b) Find the inverse of A  
# solve(A) ---->>> It shows an error  
  
# (c) Eigen values of A   
eigen(A)

## eigen() decomposition  
## $values  
## [1] 1.924695e+01 -1.246951e+00 9.342684e-17  
##   
## $vectors  
## [,1] [,2] [,3]  
## [1,] 0.1937289 0.77433837 0.4082483  
## [2,] 0.5145185 0.07288936 -0.8164966  
## [3,] 0.8353082 -0.62855964 0.4082483

# (ii) Second Matrix (B)  
B <- matrix(c(7, 5, 2, 4, 6, 7, 1, 7, 3), 3, 3)  
  
A + B

## [,1] [,2] [,3]  
## [1,] 8 6 4  
## [2,] 10 12 14  
## [3,] 11 17 14

A - B

## [,1] [,2] [,3]  
## [1,] -6 -2 2  
## [2,] 0 0 0  
## [3,] 7 3 8

# (b) A\*B  
A %\*% B

## [,1] [,2] [,3]  
## [1,] 23 37 24  
## [2,] 79 105 68  
## [3,] 135 173 112

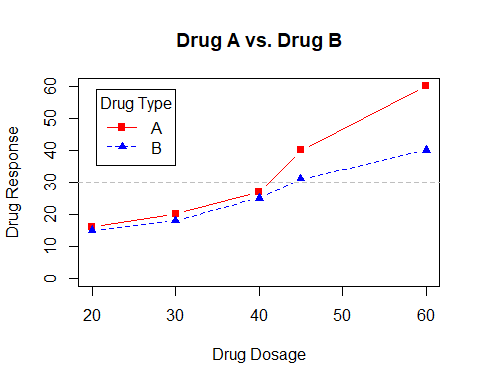
#### Problem 4

##### Consider the simple fictitious dataset that describe patient responses to two drugs at five dosage level as given below.

| Dosage | Response to Drug A | Response to Drug B |
| --- | --- | --- |
| 20 | 16 | 15 |
| 30 | 20 | 18 |
| 40 | 27 | 25 |
| 45 | 40 | 31 |
| 60 | 60 | 40 |

##### Plot response of the two drugs against dosage. [Add title, legend, text of axes, and set of a reference line at 30]

dosage <- c(20, 30, 40, 45, 60)  
drugA <- c(16, 20, 27, 40, 60)  
drugB <- c(15, 18, 25, 31, 40)  
  
plot(dosage, drugA, type="b", pch=15, lty=1, col="red",  
 ylim = c(0, 60), main = "Drug A vs. Drug B",   
 xlab ="Drug Dosage", ylab = "Drug Response")  
lines(dosage, drugB, type = "b", pch=17, lty=2, col="blue")  
abline(h=c(30), lwd=1.5, lty=2, col="gray")  
legend("topleft", inset = 0.05, title = "Drug Type", c("A", "B"), lty=c(1, 2), col = c("red", "blue"), pch = c(15, 17))



#### Problem 5

##### (i) Use Newton’s method for finding root:

##### (ii) Solve the quadratic equation using if-else() and repeat()

##### (iii) Write a function to compute n! and hence find 23!

##### (iv) Compute coefficient of variance of the dataset c(5, 6, 2, 8, 5, 9, 2, 4, 8, 3, 1, 9)

# (i) Newton's method for finding root  
f <- function(x) x^3 + 2\*x^2 - 7  
f. <- function(x) 3\*x^2 + 4\*x  
  
  
calc\_root <- function(x=1, tolerance = 0.0000001) {  
 while(abs(f(x)) > tolerance) {  
 x <- x - f(x) / f.(x)  
 }  
 return(x)  
}  
  
tolerance = 0.0000001  
calc\_root(x=1, tolerance)

## [1] 1.428818

# Solve the quadratic equation using if-else (x^2 + 2\*x-7=0)  
a = 1; b = 2; c = -7  
root\_1 = root\_2 = NULL  
dis = b^2 - 4\*a\*c  
  
if(dis < 0) {  
 cat("Roots are imaginary\n")  
 dis = dis + 0i # converting into complex form  
}  
  
root\_1 = (-b + sqrt(dis))/(2\*a)  
root\_2 = (-b - sqrt(dis))/(2\*a)  
  
cat("The roots are", round(root\_1, 2), "and", round(root\_2, 2), "\n")

## The roots are 1.83 and -3.83

# Solve the function using repeat()  
g <- function(x) x^2 + 2\*x - 7  
g. <- function(x) 2\*x + 2  
calc\_root2 <- function(x=1, tolerance = 0.0000001) {  
 repeat{  
 if(abs(g(x)) < tolerance) {  
 break  
 }  
 x <- x - g(x)/g.(x)   
 }  
 return(x)  
}  
  
root\_1 = calc\_root2(x=1, tolerance)  
root\_2 = calc\_root2(x=-4, tolerance)  
  
cat("The roots are", round(root\_1, 2), "and", round(root\_2, 2), "\n")

## The roots are 1.83 and -3.83

# (iv) Compute coefficient of variance  
y <- c(5, 6, 2, 8, 5, 9, 2, 4, 8, 3, 1, 9)  
  
cv = sd(y) / mean(y)  
cat("Coefficient of variation is ", cv)

## Coefficient of variation is 0.5525973

#### Problem 6

##### Access the default data file “stackloss” in R and fit a simple linear regression of stack.loss on Air.Flow and fitted regression line together and interpret.

head(stackloss)

## Air.Flow Water.Temp Acid.Conc. stack.loss  
## 1 80 27 89 42  
## 2 80 27 88 37  
## 3 75 25 90 37  
## 4 62 24 87 28  
## 5 62 22 87 18  
## 6 62 23 87 18

attach(stackloss)

## The following object is masked from package:datasets:  
##   
## stack.loss

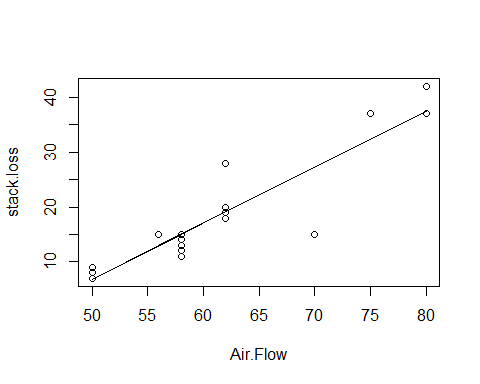
# stack.loss = a + b\*Air.Flow  
fit1 <- lm(stack.loss ~ Air.Flow)  
fit1

##   
## Call:  
## lm(formula = stack.loss ~ Air.Flow)  
##   
## Coefficients:  
## (Intercept) Air.Flow   
## -44.13 1.02

summary(fit1)

##   
## Call:  
## lm(formula = stack.loss ~ Air.Flow)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -12.2896 -1.1272 -0.0459 1.1166 8.8728   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -44.13202 6.10586 -7.228 7.31e-07 \*\*\*  
## Air.Flow 1.02031 0.09995 10.208 3.77e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.098 on 19 degrees of freedom  
## Multiple R-squared: 0.8458, Adjusted R-squared: 0.8377   
## F-statistic: 104.2 on 1 and 19 DF, p-value: 3.774e-09

plot(Air.Flow, stack.loss)  
lines(Air.Flow, fitted(fit1))



#### Problem 7

##### Access the default data file “stackloss” in R and fit a simple linear regression of stack.loss on Water.Temp and fitted regression line together and interpret.

head(stackloss)

## Air.Flow Water.Temp Acid.Conc. stack.loss  
## 1 80 27 89 42  
## 2 80 27 88 37  
## 3 75 25 90 37  
## 4 62 24 87 28  
## 5 62 22 87 18  
## 6 62 23 87 18

attach(stackloss)

## The following objects are masked from stackloss (pos = 3):  
##   
## Acid.Conc., Air.Flow, stack.loss, Water.Temp

## The following object is masked from package:datasets:  
##   
## stack.loss

# stack.loss = a + b\*Water.Temp  
fit2 <- lm(stack.loss ~ Water.Temp)  
fit2

##   
## Call:  
## lm(formula = stack.loss ~ Water.Temp)  
##   
## Coefficients:  
## (Intercept) Water.Temp   
## -41.911 2.817

summary(fit2)

##   
## Call:  
## lm(formula = stack.loss ~ Water.Temp)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7.8904 -3.6206 0.3794 2.8398 8.4747   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -41.9109 7.6056 -5.511 2.58e-05 \*\*\*  
## Water.Temp 2.8174 0.3567 7.898 2.03e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.043 on 19 degrees of freedom  
## Multiple R-squared: 0.7665, Adjusted R-squared: 0.7542   
## F-statistic: 62.37 on 1 and 19 DF, p-value: 2.028e-07

plot(Water.Temp, stack.loss)  
lines(Water.Temp, fitted(fit2))

