Homework 6

Chad Huntebrinker

2024-10-05

# Question 1:

Debug the following function that calculates the factorial of a number. Identify and fix the issue causing the function to return incorrect results for negative input values.

# Buggy Function  
get\_factorial <- function(n) {  
 if (n == 0) {  
 return(1) # Clue: Buggy location, Control the statement here  
 } else {  
 return(n \* factorial(n - 1))   
 }  
}  
  
get\_factorial(3) # works well

## [1] 6

get\_factorial(-3) # throws error

## Warning in gamma(x + 1): NaNs produced

## [1] NaN

## Answer for Question 1.

get\_factorial <- function(n) {  
 if (n <= 0) { #Changed this to <= instead of ==  
 return(1)   
 } else {  
 return(n \* factorial(n - 1))   
 }  
}  
get\_factorial(3)

## [1] 6

get\_factorial(-3)

## [1] 1

# Question 2:

You are given a function named calculate\_area\_circle intended to calculate the area of a circle. However, the function seems to have some errors. Your task is to debug the function step by step and make the necessary modifications to ensure it calculates the area correctly. Start by running the function calculate\_area\_circle with some sample inputs and observe the output. Identify any errors or unexpected behavior. Debug the code of the calculate\_area\_circle function line by line. Look for potential sources of error such as typos, incorrect variable names, or mathematical errors. After making modifications to the function, test it again with various input values to ensure that it produces the expected output for a range of scenarios. Add comments and documentation to the function code to explain its purpose, inputs, outputs, and any modifications made during debugging.

# Buggy Function: calculate\_area\_circle  
  
calculate\_area\_circle <- function(radius) {  
 # Incorrect formula: area = pi \* radius (missing exponent)  
 area <- pi \* radius  
 # Missing return statement  
 print(area) # Extra print statement  
}  
  
calculate\_area\_circle(6) # the output is is not valid

## [1] 18.84956

# it should throw the following warning messages when the function is used with undesirable arguments.   
calculate\_area\_circle(0) # Radius should be a non-zero numeric value.

## [1] 0

calculate\_area\_circle(-1) # Radius cannot be negative.

## [1] -3.141593

# calculate\_area\_circle(3,4) # Radius should be a single value.  
# calculate\_area\_circle("three") # it should produce warning message: "Only numeric values are allowed"  
# the above two lines 66 and 67 should be kept in comments while knitting, or else it will throw error while knitting to word

##Answer to Question 2

# Corrected Function: calculate\_area\_circle  
calculate\_area\_circle <- function(radius) {  
 #Added the following if statements for edge cases like if radius is a string, 0, a negative number, or has multiple values  
 if (length(radius) != 1) {  
 return("Radius should be a single value.")  
 }  
 else if(is.character(radius)){  
 return("Only numeric values are allowed")  
 }  
 else if(radius == 0) {  
 return("Radius should be a non-zero numeric value.")  
 }  
 else if(radius < 0) {  
 return("Radius cannot be negative.")  
 }  
 # Incorrect formula, was missing an exponent. Squared the radius  
 area <- pi \* radius^2  
 # Added a return statement and removed the extra print statement  
 return(area)  
}  
  
calculate\_area\_circle(6) # the output is is not valid

## [1] 113.0973

# it should throw the following warning messages when the function is used with undesirable arguments.   
calculate\_area\_circle(0) # Radius should be a non-zero numeric value.

## [1] "Radius should be a non-zero numeric value."

calculate\_area\_circle(-1) # Radius cannot be negative.

## [1] "Radius cannot be negative."

#Changed the following to a vector rather than two separate values.  
calculate\_area\_circle(c(3,4)) # Radius should be a single value.

## [1] "Radius should be a single value."

calculate\_area\_circle("three") # it should produce warning message: "Only numeric values are allowed"

## [1] "Only numeric values are allowed"

# the above two lines should be kept in comments while knitting, or else it will throw error while knitting to word

# Question 3

the matrix expression\_data contains expression values for five genes across five samples. Following to this there is a function to calculate the average expression level of a gene across multiple samples. There are some bugs in the function that causes it to return incorrect results. We can use the debug() function to set a debugging flag at the beginning of the function to help us identify the bug.

# Create example gene expression data  
gene\_names <- c("GeneA", "GeneB", "GeneC", "GeneD", "GeneE")  
sample\_names <- c("Sample1", "Sample2", "Sample3", "Sample4", "Sample5")  
  
# Generate random expression values  
set.seed(123) # for reproducibility  
expression\_data <- matrix(data = round(runif(length(gene\_names) \* length(sample\_names), min = 0, max = 100), 2),  
 nrow = length(gene\_names),  
 ncol = length(sample\_names),  
 dimnames = list(gene\_names, sample\_names))  
  
# Display the gene expression data  
expression\_data

## Sample1 Sample2 Sample3 Sample4 Sample5  
## GeneA 28.76 4.56 95.68 89.98 88.95  
## GeneB 78.83 52.81 45.33 24.61 69.28  
## GeneC 40.90 89.24 67.76 4.21 64.05  
## GeneD 88.30 55.14 57.26 32.79 99.43  
## GeneE 94.05 45.66 10.29 95.45 65.57

# Buggy Function to   
# Buggy Function  
calculate\_gene\_average <- function(expression\_data, gene\_name) {  
  
 gene\_expression <- expression\_data[gene\_name, ]  
   
 tot\_expression <- sum(gene\_expression) / 100 #Why is this line divided by 100  
   
 average\_expression <- mean(tot\_expression) #Why do we need this line?  
  
 average\_expression <- expression\_data[gene\_name + 1, ] #This line is causing the error  
   
 return(average\_expression)  
}  
  
#calculate\_gene\_average(expression\_data, "GeneA") # this is generating error  
# the above line is an erroneous code of line. it will be executed and produce some out put when you run the code in the chunk. But when knitting it will halt the knitting process. Commenting it solves the problem

## Answer to Question 3

# Corrected Function  
# Create example gene expression data  
gene\_names <- c("GeneA", "GeneB", "GeneC", "GeneD", "GeneE")  
sample\_names <- c("Sample1", "Sample2", "Sample3", "Sample4", "Sample5")  
  
# Generate random expression values  
set.seed(123) # for reproducibility  
expression\_data <- matrix(data = round(runif(length(gene\_names) \* length(sample\_names), min = 0, max = 100), 2),  
 nrow = length(gene\_names),  
 ncol = length(sample\_names),  
 dimnames = list(gene\_names, sample\_names))  
  
# Display the gene expression data  
expression\_data

## Sample1 Sample2 Sample3 Sample4 Sample5  
## GeneA 28.76 4.56 95.68 89.98 88.95  
## GeneB 78.83 52.81 45.33 24.61 69.28  
## GeneC 40.90 89.24 67.76 4.21 64.05  
## GeneD 88.30 55.14 57.26 32.79 99.43  
## GeneE 94.05 45.66 10.29 95.45 65.57

# Buggy Function. Since we're calculating the mean of a expression for a specific gene through 5 different samples, all we need to do is alter the function a bit.  
calculate\_gene\_average <- function(expression\_data, gene\_name) {  
 gene\_expression <- expression\_data[gene\_name, ]  
   
 #Changed the sum to be divided by 100 to 5 (since we're getting the average)  
 average\_expression <- sum(gene\_expression) / 5  
   
 #We can remove the other two lines as they weren't doing anything for us  
   
 return(average\_expression)  
}  
  
calculate\_gene\_average(expression\_data, "GeneA") # this is generating error

## [1] 61.586

# the above line is an erroneous code of line. it will be executed and produce some out put when you run the code in the chunk. But when knitting it will halt the knitting process. Commenting it solves the problem

# Question 4

Here we are creating a simple clinical data management example with a dataframe containing information about patients. we have created a function that calculates the average age of patients with a given diagnosis,but with some bugs. debug the function and correct it.

# Generate patient data  
patient\_data <- data.frame(  
 Patient\_id = c(1, 2, 3, 4, 5),  
 Age = c(25, 30, 45, 60, 35),  
 Gender = c("Male", "Female", "Male", "Female", "Male"),  
 Diagnosis = c("Hypertension", "Diabetes", "Hypertension", "Asthma", "Diabetes"),  
 Treatment = c("Medication", "Insulin", "Medication", "Inhaler", "Insulin")  
)  
  
# Display the patient data  
patient\_data

## Patient\_id Age Gender Diagnosis Treatment  
## 1 1 25 Male Hypertension Medication  
## 2 2 30 Female Diabetes Insulin  
## 3 3 45 Male Hypertension Medication  
## 4 4 60 Female Asthma Inhaler  
## 5 5 35 Male Diabetes Insulin

# Buggy Function  
calculate\_avg\_age <- function(data, diagnosis) {  
  
 patients <- data[Diagnosis == diagnosis, ]   
   
 avg\_age <- sum(patients$Age) / length(patients$Age)  
   
 retunr(avg\_age)  
   
 patients <- data[Diagnos == diagnosis, ]   
   
 avg\_age <- sum(patients$Age)  
   
 return(avg\_age)  
}  
  
# Test the buggy function  
#calculate\_avg\_age(patient\_data, "Diabetes")  
# the above line is an erroneous code of line. it will be executed and produce some out put when you run the code in the chunk. But when knitting it will halt the knitting process. Commenting it solves the problem

## Answer to Question 4

# Corrected Function  
  
# Generate patient data  
patient\_data <- data.frame(  
 Patient\_id = c(1, 2, 3, 4, 5),  
 Age = c(25, 30, 45, 60, 35),  
 Gender = c("Male", "Female", "Male", "Female", "Male"),  
 Diagnosis = c("Hypertension", "Diabetes", "Hypertension", "Asthma", "Diabetes"),  
 Treatment = c("Medication", "Insulin", "Medication", "Inhaler", "Insulin")  
)  
  
# Display the patient data  
patient\_data

## Patient\_id Age Gender Diagnosis Treatment  
## 1 1 25 Male Hypertension Medication  
## 2 2 30 Female Diabetes Insulin  
## 3 3 45 Male Hypertension Medication  
## 4 4 60 Female Asthma Inhaler  
## 5 5 35 Male Diabetes Insulin

# Buggy Function. Removed redundant lines and added data$ to line 230  
calculate\_avg\_age <- function(data, diagnosis) {  
 patients <- data[data$Diagnosis == diagnosis, ]   
   
 avg\_age <- sum(patients$Age) / length(patients$Age)  
   
 return(avg\_age)  
}  
  
# Test the buggy function  
calculate\_avg\_age(patient\_data, "Diabetes")

## [1] 32.5

# the above line is an erroneous code of line. it will be executed and produce some out put when you run the code in the chunk. But when knitting it will halt the knitting process. Commenting it solves the problem