Rproject2

Your Name

2024-03-12

Task 1: quamean.

```
# Define the quamean function
quamean = function(v){
  qmean = sqrt(sum(v^2)/length(v))
  return(qmean)
}

# Test the quamean function
v = c(2,23,3456,1,4,6)

quamean(v)
```

```
## [1] 1410.941
```

Task 2 :testing the functions

```
# define the Geomean
geomean = function(v){
  gmean = (prod(v))^(1/length(v))
  return(gmean)
}




# define the harmonic mean

harmean = function(v){
  a = 1/v
  hmean = length(v)/(sum(a))
  return(hmean)
}

# Set seed for reproducibility
set.seed(123)
v <- sample(1:100, 100, replace = TRUE)
```

```r
# Example usage of each function
geomean_result <- geomean(v)
harmean_result <- harmean(v)
quamean_result <- quamean(v)

# Print the results
cat("Geometric Mean:", geomean_result, "\n")
```

```
## Geometric Mean: 41.23877
```

```r
cat("Harmonic Mean:", harmean_result, "\n")
```

```
## Harmonic Mean: 28.54434
```

```r
cat("Quadratic Mean:", quamean_result, "\n")
```

```
## Quadratic Mean: 59.87996
```

Task 3: Creating Binary Vector Based on Threshold Value

```r
# define the Intermediate function

Intermediate = function(vec1, cutoff1, cutoff2){
  vsub = vec1[vec1>cutoff1 & vec1<cutoff2]
  return(vsub)

}


# Given vector
vec1 = c(2, 2, 6, 3, 546, 2346, 22, 34, 7, 21, 4)

# Cutoff values
cutoff1 = 20
cutoff2 = 1000

# Find elements between cutoff1 and cutoff2
vsub_task = Intermediate(vec1, cutoff1, cutoff2)

# Print the result
vsub_task
```

```
## [1] 546  22  34  21
```

Task 4: Conditional statement(if else)

```r
# Given vector
v1 = c(2, 36, 83, 2, 5, 7, 2, 9, 3, 12)

# Create vector v2 using ifelse
v2 = ifelse(v1 >= 5, 0, 1)

# Print the result
print(v2)
```

```
##  [1] 1 0 0 1 0 0 1 0 1 0
```

Task 5: Adjusting Temperature Labels for Newfoundland(Cold vs. Hot Classification)

```r
# Load the MASS package
library(MASS)

# Load the whiteside dataset
data(whiteside)


# Create a column for temperatures below 0 labelled cold, and above 0 labelled hot
whiteside$hotcold_new = ifelse(whiteside$Temp < 0, "cold", "hot")

# Print the entire vector
whiteside$hotcold_new
```

```
##  [1] "cold" "cold" "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"
## [11] "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"
## [21] "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "cold" "hot"  "hot"  "hot"
## [31] "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"
## [41] "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"  "hot"
## [51] "hot"  "hot"  "hot"  "hot"  "hot"  "hot"
```

Task 6: Create A vector of prime numbers between 50 and 150

```r
# Set a CRAN mirror
options(repos = c(CRAN = "https://cran.rstudio.com"))
# Installing the schoolmath package
 install.packages("schoolmath")
```

```
## Installing package into 'C:/Users/Admin/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'schoolmath' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Admin\AppData\Local\Temp\Rtmpcn6LrC\downloaded_packages
```

```r
# Load the schoolmath package
library(schoolmath)

# Define the range
a = 50
b = 150

# Initialize an empty vector to store prime numbers
prime_numbers = c()

# Loop through the range and check for prime numbers
for (i in a:b) {
  if (is.prim(i)) {
    prime_numbers = c(prime_numbers, i)
  }
}

# Print the vector of prime numbers
prime_numbers
```

```
##  [1]  53  59  61  67  71  73  79  83  89  97 101 103 107 109 113 127 131 137 139
## [20] 149
```

Task 7:Using vectorization to find the sum of a vector equal to the quotient of vectors .

```r
# Given vectors
v1 = c(72, 3, 57, 2, 8, 24, 7)
v2 = c(7, 24, 8, 2, 57, 3, 72)

# Calculate the quotient vector using vectorization
quotient_vector = v1 / v2

# Sum of the quotient vector
sum_of_quotient = sum(quotient_vector)

# Print the result
sum_of_quotient
```

```
## [1] 26.77329
```

Task 8: 25 prime numbers following 101

```r
# Load the schoolmath package
library(schoolmath)

# Function to find the next n prime numbers following a given start number
```

```r
nextprimes = function(startnumber, numprimes) {
  primi = 0
  i = startnumber + 1
  primevec = c()

  while (primi < numprimes) {
    if (is.prim(i)) {
      primi = primi + 1
      primevec = c(primevec, i)
    }
    i = i + 1
  }

  return(print(primevec))
}

# Find the next 25 prime numbers following 101
nextprimes(101, 25)
```

```
##  [1] 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197
## [20] 199 211 223 227 229 233
```