Stat 291 - Recitation 8

Orçun Oltulu

17 / 12 / 2021

Functions:

Exercise 1:

Write an R function which takes a value and returns its square. Then test it with some values.

```r
square_function <- function(x){
  x^2
}
```

```r
for(x in c(2,12,-4,0.5)){
  square_function(x)
}
```

Exercise 2:

Part A:

Write an R function which prints your name for a given number of time.

```r
print_my_name <- function(x){
  for(i in 1:x){
    print("OrcunOltulu")
  }
}
```

```r
print_my_name(4)
```

```
## [1] "OrcunOltulu"
## [1] "OrcunOltulu"
## [1] "OrcunOltulu"
## [1] "OrcunOltulu"
```

Part B:

Now, write a function that again prints a string for a given number of time. This time the string will be given by user, but if the user does not want to give any string, it will print your name by default.

e.g. "a <- function(x, string = 'yournamehere'){}"

```r
print_my_name2 <- function(x, string = "OrcunOltulu"){
  for(i in 1:x){
    print(string)
  }
}
```

```r
print_my_name2(3, string = "STAT291")
```

```
## [1] "STAT291"
## [1] "STAT291"
## [1] "STAT291"
```

Exercise 3:

Write an R function which calculates the Euclidean distance between two points.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

```r
dist_two_points <- function(x,y){
  sqrt(sum((x-y)^2))
}
```

```r
dist_two_points(x = c(1,2,3), y = c(-1,0,2))
```

```
## [1] 3
```

Exercise 4:

Write an R function which takes 2 integers and an operation (addition, subtraction, multiplication and division), then returns a result of the mathematical operation.

e.g. myfunction(2,4,"multiplication") -> 8

Also, prints an error message for undefined operations.

```r
# alternatively,
# If a function cannot complete its assigned task,
# it should throw an error with stop(),
# which immediately terminates the execution of the function.
```

```r
mycalculator <- function(a, b, operation){
  result <- switch(operation,
                   "addition" = a + b,
                   "subtraction" = a - b,
                   "multiplication" = a * b,
                   "division" = a / b,
                   stop(paste("no such operation:", operation)))
  return(result)
}
```

```r
mycalculator(a = 12, b = 4, operation = "addition")
```

```
## [1] 16
```

```r
mycalculator(a = 12, b = 4, operation = "division")
```

```
## [1] 3
```

```r
mycalculator(7,8,"power")
```

```
## Error in mycalculator(7, 8, "power"): no such operation: power
```

Exercise 5:

Write an R function which returns median of a vector. (without using median() function)

$$Med(x) = \begin{cases} \frac{X_{[\frac{n}{2}]} + X_{[\frac{n}{2}+1]}}{2} & x \text{ is even} \\ X_{[\frac{n+1}{2}]} & x \text{ is odd} \end{cases}$$

```r
median_function <- function(x){
  size <- length(x)
  x_sort <- sort(x)
  if(size %% 2 == 0){
    med <- (x_sort[size/2] + x_sort[size/2 + 1]) / 2
  } else{
    med <- x_sort[(size+1) / 2]
  }
  return(med)
}
```

```r
set.seed(291)
x1 <- sample(100, size = 14)
x2 <- sample(100, size = 15)

# sort(x1) # to find median manually
# sort(x2) # to find median manually

median_function(x1)
```

```
## [1] 42.5
```

```
median_function(x2)
```

```
## [1] 60
```

Exercise 6:

Part A.

Write an R function which converts Celsius into Fahrenheit and find the boiling temperature of water.

(**Hint:** Fahreneit = Celsius * 1.8 + 32)

```
temperature_converter <- function(celsius){
  fahrenheit <- celsius * 1.8 + 32
  return(fahrenheit)
}
```

```
paste("Water boils at",
      temperature_converter(100), "Fahrenheit")
```

```
## [1] "Water boils at 212 Fahrenheit"
```

Part B.

Add some new commands so that your function also converts celsius to kelvin and rankine and returns a **'list'** of 4 different temperature measures (Celsius, Fahrenheit, Kelvin, Rankine)

(**Hint:** Kelvin = Celsius + 273.15, Rankine = (Celsius + 273.15) * 9/5)

```
temperature_converter <- function(celsius){

  fahrenheit <- celsius * 1.8 + 32
  kelvin <- celsius + 273.15
  rankine <- (celsius + 273.15) * 9/5

  out <- list(Celsius = celsius,
              Fahrenheit = fahrenheit,
              Kelvin = kelvin,
              Rankine = rankine)
```

```r
  return(out)
}
```

```r
temperature_converter(100)
```

```
## $Celsius
## [1] 100
##
## $Fahrenheit
## [1] 212
##
## $Kelvin
## [1] 373.15
##
## $Rankine
## [1] 671.67
```

Exercise 7:

Write a R program to find the factors of a given number.

```r
print_factors = function(n) {
  factors <- numeric()
  for(i in 1:n) {
    if((n %% i) == 0) {
      factors <- c(factors,i)
    }
  }
  cat("There are", length(factors), "factors of", n,
      "which are \n", factors, "\n")
}

print_factors(4)
```

```
## There are 3 factors of 4 which are
##  1 2 4
```

```r
print_factors(7)
```

```
## There are 2 factors of 7 which are
##  1 7
```

```r
print_factors(12)
```

```
## There are 6 factors of 12 which are
##  1 2 3 4 6 12
```

Exercise 8:

Write an R function which constructs a Confidence Interval for a Normally Distributed random variable.

Hint 1: Your function must take a sample vector and a confidence level then construct C.I.

Hint 2:

$$CI = \bar{x} \pm Z_{\frac{\alpha}{2}} * \frac{\sigma}{n}$$

Hint 3: qnorm(alpha/2) -> $Z_{\frac{\alpha}{2}}$

```r
CI_function <- function(x, conf.level = 0.95){
  alpha <- 1 - conf.level
  lower <- mean(x) - qnorm(1 - alpha/2) * (sd(x) / length(x))
  upper <- mean(x) + qnorm(1 - alpha/2) * (sd(x) / length(x))
  out <- c(lower, upper)
  names(out) <- paste(c("lower ","upper "), "CI ", 1-alpha/2,"%", sep="")
  return(out)
}
```

```r
set.seed(291)
mysample <- rnorm(20, mean = 15, sd = 4)

CI_function(mysample)
```

```
## lower CI 0.975% upper CI 0.975%
##        14.77397        15.74292
```