Carson Slater

6375 Homework 2 Solutions

```
# Sourcing my helpers.R file
source("/Users/carson_slater1/Documents/Fall2023/Baylor_Statistics_HW/Computation I/helpers.R")
```

Question 1

I have read Trefethen and Bau Lectures 2 and 3.

Also, I did not use any artificial intelligences or resources other than my `helpers.R` file, Trefethen and Bau, Advanced R, this tutorial on the Quarto website to help render errors in code, and base `R`.

Question 2

```
ip <- function (x, y) {
  dot(Conj(x), y)
}
```

```
# Testing ip()
x <- c(1,  1)
y <- c(1, 1i)

ip(x, y)
# [1] 1+1i

a <- 1i

ip(a*x, y)
# [1] 1-1i
```

Question 3

```
norm <- function(x) {
  if (is.vector(x) == FALSE & is.numeric(x) == TRUE) {
    stop("`norm()` only takes vectors as inputs")
  }
```

```
  else {
    sqrt(dot(Conj(x), x))
  }
}
```

```
# Testing norm()
c(1,1) |> norm()
# [1] 1.414214

1:3 |> norm()
# [1] 3.741657

c(1,1i) |> norm()
# [1] 1.414214+0i
```

Question 4

```
is.zero_vec <- function(x) {

  # Innocent until guilty
  result <- FALSE

  if (norm(x) <= sqrt(.Machine$double.eps)) {
    result <- TRUE
  }
  result
}

normalize <- function(x) {
  if (is.zero_vec(x) == TRUE) {
    stop("Input ‘x‘ is numerically equivalent to the zero vector.")
  }
  x/norm(x)
}
```

```
# Checking normalize()
c(1, 1) |> normalize()
# [1] 0.7071068 0.7071068
```

```r
1:3 |> normalize()
# [1] 0.2672612 0.5345225 0.8017837

c(0, 0, 0, 0) |> normalize()
# Error in normalize(c(0, 0, 0, 0)): Input ‘x‘ is numerically equivalent to the zero vector.
```

Question 5

```r
angle <- function(x, y) {
  if (length(x) != length(y)) {
    stop("Vectors do not possess same number of dimensions.")
  }
  as.numeric(acos(ip(x, y)/(norm(x)*norm(y))))
}
```

```r
# Testing angle()
angle(c(1,0), c(1,1))
# [1] 0.7853982

pi/4
# [1] 0.7853982

angle(c(1,0), c(1,1i))
# [1] 0.7853982
```

Question 6

```r
bar <- function(x) {
  if (is.vector(x) == FALSE | is.numeric(x) == FALSE){
    stop("Input must be a numeric vector.")
  }
  total(x)/length(x)
}
```

```r
# Testing bar()
bar(1:7)
# [1] 4
```

```r
mean(1:7)
# [1] 4
```

Question 7

```r
center <- function(x) {
  # I initialize `avg` so bar() does not
  # run every time the for loop executes
  avg <- bar(x)
  centered <- vector()
  for (i in 1:length(x)) {
    centered[i] <- x[i] - avg
  }
  centered
}
```

```r
# Testing center()
center(1:3)
# [1] -1  0  1
```

Question 8

```r
corr <- function(x, y) {
  cos(angle(center(x), center(y)))
}
```

```r
# Checking corr
corr(c(1,2,3), c(1,2,2))
# [1] 0.8660254

cor(c(1,2,3), c(1,2,2))
# [1] 0.8660254
```

Question 9

```
lp_norm <- function(x, p = 2) {
  if (p < 1) {
    stop("‘lp_norm()‘ is only defined to work with ‘p‘ >= 1.")
  }
  if (p == 1) {
    my_norm <- total(abs(x))
  }
  if (p == Inf) {
    my_norm <- max(abs(x))
  } else {
    my_norm <- (total(abs(x)^p))^(1/p)
  }
  my_norm
}
```

```
# Testing lp_norm()
x <- c(3, 4)
lp_norm(x)
# [1] 5

lp_norm(x, p = 1)
# [1] 7

lp_norm(x, p = Inf)
# [1] 4

lp_norm(x, p = 1/2)
# Error in lp_norm(x, p = 1/2): ‘lp_norm()‘ is only defined to work with ‘p‘ >= 1.
```

Question 10

```
mat_norm <- function (A, type = "o") {

  # Initializing variables
  row_norms <- vector()
  col_norms <- vector()

  if (is.matrix(A) == FALSE) {
```

```
    stop("‘mat_norm()‘ only accepts an array or matrix object.")
  }
  if (type == "o") {
    for (i in 1:ncol(A)) {
      col_norms[i] <- lp_norm(A[, i], p = 1)
    }
    my_norm <- max(col_norms)
  } else if (type == "i") {
    for (i in 1:nrow(A)) {
      row_norms[i] <- lp_norm(A[i, ], p = 1)
    }
    my_norm <- max(row_norms)
  } else if (type == "f") {
    my_norm <- lp_norm(as.vector(A), p = 2)
  } else {
    stop("‘type‘ must be ‘o‘ for one, ‘f‘ for Frobenius, or ‘i‘ for infinity.")
  }
  my_norm
}
```

```
# Testing mat_norm()
(A <- matrix(1:4, nrow = 2, byrow = TRUE))
#      [,1] [,2]
# [1,]    1    2
# [2,]    3    4

mat_norm(A)
# [1] 6

mat_norm(A, type = "o")
# [1] 6

base::norm(A, type = "o")
# [1] 6

mat_norm(A, type = "i")
# [1] 7

base::norm(A, type = "i")
# [1] 7
```

```r
mat_norm(A, type = "f")
# [1] 5.477226

base::norm(A, type = "f")
# [1] 5.477226
```