Homework 1

Michael Zheng

2/3/2021

Task 1

```r
mh_distance <- function(x, y) {
  if(is.na(x) | is.na(y) | is.infinite(x) | is.infinite(y))  {
    warning('x and y cannot be NA, NaN, Inf, or -Inf')
    return(-1)
  } else if(mode(x) != "numeric" &
     mode(x) != "character" &
     mode(x) != "logical" &
     mode(y) != 'numeric' &
     mode(y) != "character" &
     mode(y) != "logical") {
    warning('x and y are not of type logical, character, or numeric')
    return(-1)
  } else if(typeof(x) != typeof(y)) {
    warning('x and y are not the same type')
    return(-1)
  } else if (is.logical(x)) {
    if(x == y) {
      return(0)
    } else {
      return(1)
    }
  } else if (is.numeric(x) | is.numeric(y)) {
    if(x %% 1 != 0 | y %% 1 != 0) {
      warning('x or y contains decimal values')
      return(-1)
    } else {
      x <- abs(x)
      y <- abs(y)
    }
  }
    x <- as.character(x)
    y <- as.character(y)
    l <- nchar(x)
    if(nchar(x) != nchar(y)) {
      warning('x and y are not the same length')
      return(-1)
    } else {
      x_split <- unlist(strsplit(x, split = ""))
      y_split <- unlist(strsplit(y, split = ""))
      a <- 1
      b <- 0
      while (a <= l) {
        if(x_split[a] != y_split[a]) {
          b <- b + 1
        }
        a <- a + 1
      }
      return(b)
    }

}
```

Task 2

Initial test cases (return a non-negative modified Hamming distance)

```
mh_distance(x = "abc", y = "abc")
```

```
[1] 0
```

```
mh_distance(x = T, y = FALSE)
```

```
[1] 1
```

```
mh_distance(x = "523890", y = "752839")
```

```
[1] 5
```

```
mh_distance(x = 2341, y = 2350)
```

```
[1] 2
```

Added test cases that return a non-negative modified Hamming distance result.

```
mh_distance(x = TRUE, y = TRUE)
```

```
[1] 0
```

```
mh_distance(x = FALSE, y = F)
```

```
[1] 0
```

```
mh_distance(x = "having fun", y = "have money")
```

```
[1] 7
```

```
mh_distance(x = 54803, y = 34821)
```

```
[1] 3
```

```
mh_distance(x = -233,y = 233)
```

```
[1] 0
```

Initial test cases (return values of -1)

```
mh_distance(x = 52, y = 113)
```

```
Warning in mh_distance(x = 52, y = 113): x and y are not the same length
```

```
[1] -1
```

```r
mh_distance(x = "swimming", y = "winning")
```

```
Warning in mh_distance(x = "swimming", y = "winning"): x and y are not the same
length

[1] -1
```

```r
mh_distance(x = NA, y = TRUE)
```

```
Warning in mh_distance(x = NA, y = TRUE): x and y cannot be NA, NaN, Inf, or -
Inf

[1] -1
```

```r
mh_distance(x = 1.5, y = 2.5)
```

```
Warning in mh_distance(x = 1.5, y = 2.5): x or y contains decimal values

[1] -1
```

Added test cases that return a value of -1.

```r
mh_distance(x = 52, y = "52")
```

```
Warning in mh_distance(x = 52, y = "52"): x and y are not the same type

[1] -1
```

```r
mh_distance(x = 1.5, y = 2)
```

```
Warning in mh_distance(x = 1.5, y = 2): x or y contains decimal values

[1] -1
```

```r
mh_distance(x = 1, y = 2.4)
```

```
Warning in mh_distance(x = 1, y = 2.4): x or y contains decimal values

[1] -1
```

```r
mh_distance(x = NaN, y = -Inf)
```

```
Warning in mh_distance(x = NaN, y = -Inf): x and y cannot be NA, NaN, Inf, or -
Inf

[1] -1
```

```r
mh_distance(x = 300, y = 2.5)
```

```
Warning in mh_distance(x = 300, y = 2.5): x or y contains decimal values

[1] -1
```

Task 3

Consider the pair of vectors s and w given below.

```r
s <- c(26, 50123, 456.12, 8, 0)
w <- c(22, 50000, 451.00, 88, 0)
```

```r
x <- 1
while (x <= length(s)) {
  y <- suppressWarnings(mh_distance(x = s[x], y = w[x]))
  if(y != -1)
  print(paste0("The modified Hamming distance between ",s[x], " and ", w[x], " is ", y))
  x <- x + 1
}
```

```
[1] "The modified Hamming distance between 26 and 22 is 1"
[1] "The modified Hamming distance between 50123 and 50000 is 3"
[1] "The modified Hamming distance between 0 and 0 is 0"
```

Task 4

I first handled NA, Nan, Inf, and -Inf by using an if statement to stop x and y vectors that have those value. Then I used another if statement to stop if neither x or y wer numeric, character, or logical. Then, I checked to see if x and y are the same type and stopped if they were not. Afterwards, I checked if x or y were numeric so we can determine if x or y is a decimal or checked if x or y were boolean so I could change the boolean into a 1/0 so that it would be representative of the distance formula. For the final invalid input, I used an if statement to check to see if x and y were the same length using numchar(). Then, after filtering through all invalid outputs, I converted both x and y into character strings and then split each into a vector of separate letters using strsplit. I then compared through each element of the same order in both x and y. I had a dummy variable called b that I used to keep a count of each different between x and y. One weakness in my code is that it is somewhat long and requires evaluating x and y separately in each case. If I were able to evaluate x and y simultaneously, I may increase code efficiency.