Homework 2

Ze Yang (zey@andrew.cmu.edu)

Due Thursday, September 14 at 3:00 PM

You should submit the Rmd file with your answers in the appropriate spaces. Rename the file as `YOURANDREWID_HW2.Rmd` and submit it via Canvas. Also submit the `.pdf` file that is produced.

Of course, any code that is written should be tested. In each case you are requested to include some **simple** examples showing that your code works. Think about how your example(s) can illustrate the range of possibilities that the code could face.

1. Suppose that `charvec` is a vector of type `character`. Write a **single** line of R code that returns a new, single character string that consists of all of the elements of `charvec` concatenated end to end. For example, `charvec = c("a","b","c")` should become `"abc"`.

Solution:

```r
concat.str = function(str.vec) {
  #' Concatenate vector of strings into a new string.
  #' @param str.vec: a vector of strings.
  #' @return the concatenated string.
  return(paste(str.vec[!is.na(str.vec)], collapse = ''))
}
```

Testcases:

```r
# vanilla case, "Hello world!" expected
stopifnot(concat.str(c("Hello"," ","world!")) ==
            "Hello world!")

# we should not include NA to the concatenated string,
# if there is any. "abc ef g" expected
stopifnot(concat.str(
  c("a",NA, "b","c", NA, " ", "e", "f", " ", "g")) ==
    "abc ef g")

# when is.na() is applied to empty vector, it will give a warning
# but we can live with that, it's natural expected behavior.
# Note that concat.str returns an empty string.
stopifnot(concat.str(c()) == "")
```

```
## Warning in is.na(str.vec): is.na() applied to non-(list or vector) of type
## 'NULL'
```

```r
# when trying to concatenate an NA-vector, also return empty string
stopifnot(concat.str(c(NA, NA, NA)) == "")

# Work on integer vectors for free, "20171211" expected.
stopifnot(concat.str(c(2017, 12, 11)) == "20171211")
```

2. Write an R function that takes a numeric vector as input and returns the `five-number summary` as a **list**. The five-number summary consists of the minimum, the 25th, 50th, and 75th percentile, along with the mapximum. Name the components of the list appropriately.

Solution:

```r
percentile.list = function(vec) {
  #' Calculate the five-number summary of a numeric vector.
  #' @param vec: a numeric vector.
  #' @return list of 5-number summary.
  l.percentile = as.list(quantile(vec, na.rm = T))
  names(l.percentile) = c('min','percentile.25','median','percentile.75','max')
  return(l.percentile)
}
```

Testcases:

```r
# vanilla case
l.1 = percentile.list(c(1,2,3,4,5))
stopifnot(l.1$min == 1)
stopifnot(l.1$percentile.25 == 2)
stopifnot(l.1$median == 3)
stopifnot(l.1$percentile.75 == 4)
stopifnot(l.1$max == 5)

l.2 = percentile.list(c(4,6,10,20,43,6,8,90,22,1,0,-6,-7,-10))
stopifnot(l.2 == c(-10, 0.25, 6, 17.5, 90))

# NA handling
l.3 = percentile.list(c(1,2,3,NA,4,5,NA))
stopifnot(l.3 == c(1,2,3,4,5))

# empty and all-NA situation
l.4 = percentile.list(c(NA, NA))
l.5 = percentile.list(c())
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type 'NULL'
```

```r
stopifnot(is.na(l.4) == c(T,T,T,T,T))
stopifnot(is.na(l.5) == c(T,T,T,T,T))
```

3. Write an R function that, when given a vector of strings, returns the position(s) of the

longest strings in the vector.

Solution:

```
which.longest = function(str.vec) {
  #' Find the position(s) of longest strings in a vector.
  #' @param str.vec: a vector of strings.
  #' @return the positions.
  return(which(nchar(str.vec)==max(nchar(str.vec), na.rm=T)))
}
```

Testcases:

```
# Test cases
# Vanilla case, 3 expected
stopifnot(which.longest(c("a", "ab", "abc", "a")) == c(3))


# Multiple longest strings
# we should find all positions, 2,3,6 expected
stopifnot(which.longest(c("a", "abc", "def", "a", NA, 'ghl')) ==
            c(2, 3, 6))

# NA handling
# 4 expected
stopifnot(which.longest(c(NA, "a", "ab", "abc", NA)) == c(4))

# All-NA vector
# it should not get anything. Actually it gives a warning
# because it's trying to take maximum on all-NA vector.
# we can live with this behavior, some kind of warning is "expected".
stopifnot(which.longest(c(NA, NA)) == c())
```

```
## Warning in max(nchar(str.vec), na.rm = T): no non-missing arguments to max;
## returning -Inf
```

```
# Empty vector
# it gives a warning, as expected.
stopifnot(which.longest(c()) == c())
```

```
## Warning in max(nchar(str.vec), na.rm = T): no non-missing arguments to max;
## returning -Inf
```

```
# Spaces should be treated like chars
# 2,4 expected
stopifnot(which.longest(c("  ", "     ", "abcd", "abcde")) == c(2,4))
```

4. Create an **infix** operator that returns whether or not a number is a multiple of another. Call it %m%. In other words, `10 %m% 5` should be `TRUE` while `10 %m% 3` should be `FALSE`.

Solution:

```
'%m%' = function(vec, divisor) {
  #' calculate whether entries in \code{vec} are multiples
  #' of \code{divisor}. If \code{vec} has same dimension as
  #' \code{divisor}, the calculation is performed element-wise.
  #' Otherwise, the one with less entries will be duplicated to
  #' match the other's dimension, then perform element-wise calculation.
  #' @param vec: a vector of strings.
  #' @param divisor: a vector of strings.
  #' @return logical vector.
  return(vec %% divisor == 0)
}
```

Testcases:

```
# scalar cases
stopifnot(8 %m% 2 == T)   # 8 is multiple of 2
stopifnot(8 %m% 3 == F)   # 8 is not multiple of 3
stopifnot(is.na(8 %m% 0)) # divide 0 is not valid, return NA
stopifnot(is.na(8 %m% NA)) # NA handling
stopifnot(is.na(NA %m% 2)) # NA handling
stopifnot(is.na(NA %m% NA)) # NA handling

# vector cases
# if If vec has same dimension as divisor, the calculation is performed
# element-wise. Otherwise, the one with less entries will be duplicated to
# match the other's dimension, then perform element-wise calculation.

# case.1, same dimensions
stopifnot(c(2,4,8,16) %m% c(2,2,2,2) == c(T,T,T,T))
stopifnot(c(3,6,9,12) %m% c(2,2,2,2) == c(F,T,F,T))
stopifnot(c(3,6,9,12) %m% c(1,2,3,6) == c(T,T,T,T))
stopifnot((is.na(c(3,6,9,NA) %m% c(NA,2,0,6)) == c(T,F,T,T)))

# case.2, vec has lower dimension
stopifnot(256 %m% c(3,16,77,128) == c(F,T,F,T))
# Right broadcasting, equivalent to c(256, 256, 256, 256) %m% c(...)
stopifnot(c(77,256) %m% c(3,16,77,128) == c(F,T,T,T))
# Right broadcasting, equivalent to c(77, 256, 77, 256) %m% c(...)
stopifnot(c(256,16,77) %m% c(3,16,77,128) == c(F,T,T,T))
```

```
## Warning in vec%%divisor: longer object length is not a multiple of shorter
## object length
```

```
# Right broadcasting, equivalent to c(256, 16, 77, 256) %m% c(...)
# a warning is generated saying longer length is not multiple of the shorter length
# so broadcasting is partially done.
```

```r
# case.3, divisor has lower dimension
stopifnot(c(2,4,8,16) %m% 8 == c(F,F,T,T))
# Left broadcasting, equivalent to c(...) %m% c(8, 8, 8, 8)
stopifnot(c(2,4,8,16) %m% c(2,8) == c(T,F,T,T))
# Left broadcasting, equivalent to c(...) %m% c(2, 8, 2, 8)
stopifnot(c(2,4,8,16) %m% c(16,8,4) == c(F,F,T,T))
```

```
## Warning in vec%%divisor: longer object length is not a multiple of shorter
## object length
```

```r
# Left broadcasting, equivalent to c(...) %m% c(16, 8, 4, 16)
# a warning is generated saying longer length is not multiple of the shorter length
# so broadcasting is partially done.
```

5. Go to the following website: https://www.sec.gov/data/foiadocsfailsdatahtm and download the **July 2017, Second Half** data set. Read these data into R and create an appropriate data frame. Be sure that each column is in an appropriate form (date, factor, character, etc.) When reading in the file, **do not make any changes to the data file itself**. Resolve any issues using appropriate R commands.

Issues:

- The data is parsed with delimiter `|`.
- The last two rows are not part of data, we should skip them.
- Several columns do not have correct type. In particular:
  - `SETTLEMENT.DATE` should be date field.
  - `DESCRIPTION` had better be a char field.
  - `SYMBOL` and `CUSIP` are tickers symbols and CUSIP numbers. They may serve as keys to query the data, so we leave them as factor fields.
  - `PRICE` should be a numeric field.

Solution:

```r
df = read.delim('cnsfails201707b.txt',sep='|',
                nrow=length(readLines("cnsfails201707b.txt")) - 3)
str(df)
```

```
## 'data.frame':    47420 obs. of  6 variables:
##  $ SETTLEMENT.DATE : int  20170717 20170717 20170717 20170717 20170717 20170717 20170
##  $ CUSIP           : Factor w/ 9299 levels "000304105","000307108",..: 8918 8919 8920
##  $ SYMBOL          : Factor w/ 9313 levels "","1973R","2125REGWAY",..: 2160 2122 131
##  $ QUANTITY..FAILS.: int  40380 15936 15 500 107758 5128 428 462 318 191 ...
##  $ DESCRIPTION     : Factor w/ 8975 levels "1-800 FLOWERS.COM INC",..: 2132 2233 118
##  $ PRICE           : Factor w/ 10385 levels ".","0.01","0.02",..: 8876 2899 8399 48 2
```

```r
df$SETTLEMENT.DATE = as.Date.character(df$SETTLEMENT.DATE, format='%Y%m%d')
df$DESCRIPTION = as.character(df$DESCRIPTION)
df$PRICE = as.character(df$PRICE)
df$PRICE = replace(as.character(df$PRICE), df$PRICE==".","0.00")
```

```r
df$PRICE = as.numeric(df$PRICE)
str(df)
```

```
## 'data.frame':    47420 obs. of  6 variables:
##  $ SETTLEMENT.DATE : Date, format: "2017-07-17" "2017-07-17" ...
##  $ CUSIP           : Factor w/ 9299 levels "000304105","000307108",..: 8918 8919 8920
##  $ SYMBOL          : Factor w/ 9313 levels "","1973R","2125REGWAY",..: 2160 2122 131
##  $ QUANTITY..FAILS.: int  40380 15936 15 500 107758 5128 428 462 318 191 ...
##  $ DESCRIPTION     : chr  "DAIMLER AG" "DEUTSCHE BANK AG NAMEN AKT (DE" "ADIENT PLC O
##  $ PRICE           : num  74.4 18.86 68.85 0.47 15.25 ...
```