Notes on Elementary MapReduce Implementation for a Time Series Data frame

Reference article:
https://www.r-bloggers.com/mapreduce-with-r-on-hadoop-and-amazon-emr/

Input file:

Saved the input file using `write.table()` with no column and no row names:

```
write.table(data_frame, "file_name.csv", col.names = FALSE,
row.names = FALSE, quote=FALSE, sep="\t")
```

Added a column to keep track of the row numbers.

The Mapper:

It has to be converted to an executable file from R, so be careful with all the details. `#!` has a special meaning…

```
#! /usr/bin/env Rscript
# map.R
input <- file("stdin", "r")
while(length(x <- readLines(input, n=1, warn=FALSE)) > 0) {
      # in case of empty lines
      # more sophisticated defensive code makes sense here
      if(nchar(x) == 0) break
      y<-unlist(strsplit(x, "\t"))
      rnmb<-as.numeric(y[1])
      v<-as.numeric(y[2:length(y)])
      # find the max value
      mx<-max(v)
      # find its location in the time series
      loci<-rnmb-1+which.max(v)
     cat(loci, "\t", mx, "\n")
}
close(input)
```

The Reducer:

```
#! /usr/bin/env Rscript
# reduce.R
input <- file("stdin", "r")

# initialize variables that keep
# track of the state

is_first_line <- TRUE
loci<--8
counter<-1
while(length(line <- readLines(input, n=1, warn=FALSE)) > 0) {
      line <- unlist(strsplit(line, "\t"))

      # current line belongs to previous
      # line's key pair
      if(!is_first_line &&
         loci == line[1]) {
            counter <- counter + 1
      }
      # current line belongs either to a
      # new key pair or is first line
      else {
            # new key pair - so output the last
            # key pair's result
            if(!is_first_line) {
                  # language / 2-gram / count
                  cat(loci,"\t",counter,"\n")
                  counter<-1
            }

            # initialize state trackers
            loci <- line[1]

            is_first_line <- FALSE
      }
}

# the final record
cat(loci,"\t",counter,"\n")

close(input)
```

To Run on the Mac:

The input file(s) and both the Map.R and Reduce.R files should be all in the same folder. Navigate to that folder with the Terminal and then:

```
$ chmod 755 reduce.R
$ chmod 755 map.R
```

or

```
$ chmod 755 map.R reduce.R
```

to create the executables.

Then use **cat** to feed the input into **map.R** line by line, then sort and send input to **reduce.R:**

```
$ cat file_name.csv | ./map.R | sort | ./reduce.R
```