Problem 1(a): Since no separation format is specificed for tmp1.csv, it's saved one letter per line and "/n/r"takes 2 bytes. tmp2.csv is separated by tab and each tab takes less than 1 byte. Therefore, tmp2.csv has a smaller storage.

tmp3 is saved as Rda.file, which is more compact and can store any type of R data sctructure. Therefore, it has a smaller storage than tmp4.csv, which is a regular comman separated txt. file. tmp5.csv is rounded to 2 digits thus the total bytes are reduced since the number of bytes taken by digits is reduced.

1(b) save() saves R object and can later be read into R using load(). ASCII characters are stored as integer values, from A-Z, the integer values associated with each letter has an increasing trend. Therefore, the file tmp6.csv generated from sample() has various letters with a larger stoage. tmp7. csv has repetitve "a" letters thus it has a much smaller storage.

```
> #==============Problem 2(a) ===============
>
> #=========================NOTE==============================
> #Since the html file generated from citation_func is too long, I silenced the output
> #and returned html file as an object #instead.
> #The resulting data frame did show the existence of the generated html file.
>
>
>
> #This part mainly uses "readLines" to get html,  extract the link with citations info usi
> library(XML)
> library(RCurl)
> library(stringr)
> library(rvest)
> citation_func <- function(n){
+   #1. check the whether user input is a string, and place a "+" in the middle to prepare
+ if (is.numeric(n)) return("input should be a string")
+ if(is.na(n)==TRUE) return ("input should not be NA")
+ new_name <- gsub("\\s+", "+", n)
+
+ #2. Paste name and URL, get html file and parse it into a readible form
+ link <- paste("https://scholar.google.com/scholar?hl=en&q=+",new_name,"+&as_sdt=1%2C5&as_s
+ xData <- readLines(link)
+ file_html<-htmlParse(xData)
+
+ #3. get all the links and extract the ones containing "citations" info; generate a new cit
+ link_set <- getHTMLLinks(file_html)
+ output<-str_extract(link_set,"user=")
+ str_1 <- "https://scholar.google.com"
+ str_2 <- link_set[!is.na(output)][1] #extracted that url
+
+ #If the author does not exist, str_2 would return NA
+ if(is.na(str_2)) return("Invalid Author Name")
```

```
+
+ #4. Generate author's ID
+ s <- unlist(str_split(str_2, "="))
+ s <- unlist(str_split(s[2], "&"))
+ print(s[1])
+
+ #5. now generate citation URL and get the citation page html file
+ str_cit <- paste(str_1,str_2,sep = "")  #new_url formed
+ print(str_cit)
+ result_html<-readLines(str_cit)
+ return(result_html)
+ }
>

> #================Problem 2(b)==================
> #For part(b), the table generated from the website contains has overlapped title, author i
> #Approach1: directly get xmlVaule via xPath of the html's "div" elements -> Failed, didn't
> #Approach2: using regEx to process the overlapping info; separate and recombine them into
>
> table_func <- function(f){
+ #1. check whats generated from the htmltable; two tables, "feature_2" contains useful info
+ features_1 <- readHTMLTable(f, which = 1)
+ features_2 <- readHTMLTable(f,which = 2)
+ head(features_1)
+ head(features_2)
+
+ #2. Observation: author names and publication info are cantenated together.
+ # trivial string processing
+ t<-gsub("Mc","MC", features_2[[1]])
+ t<-gsub("science", "Science", t)
+ t<-gsub("arXiv", "ARXiv", t)
+ t<-gsub("SNE", "SNe", t)
+
+ #3. Use RegEx to find odd patterns, eg. "HintonNature 333,..."; Insert % to mark the odd p
+ t<- gsub("([[:lower:]])([[:upper:]])", "\\1%\\2", t)
+ t<- gsub("\\,\\s\\.\\.\\.", "%", t)
+
+ #4. Split strings based on % marks
+ split <- strsplit(t, "%")
+ split <- unlist(split)
+ split <- matrix(split, nrow= 3, ncol = 20)
+
+ #5. Separated title, author, jornal_info and combine into a dataframe
+ title <- split[1,]
+ author <- split[2,]
+ journal_info <- split[3,]
```

```
+ df <- data.frame(title, author, journal_info, features_2[,2], features_2[,3])
+ names(df) <-c("title", "authors", "journal", "number of citations", "publication_year")
+ head(df)
+ }
> #=================TEST Functions ========================================
> r_1 <- citation_func("Geoffrey Hinton")

[1] "JicYPdAAAAAJ"
[1] "https://scholar.google.com/citations?user=JicYPdAAAAAJ&hl=en&oe=ASCII&oi=ao"

> table_func(r_1)

                                                        title
1             Learning representations by back-propagating errors
2          Learning internal representations by error-propagation
3          Learning internal representations by error propagation
4                                Parallel distributed processing
5 Imagenet classification with deep convolutional neural networks
6                   A fast learning algorithm for deep belief nets
                                      authors
1            DE Rumelhart, GE Hinton, RJ Williams
2            DE Rumelhart, GE Hinton, RJ Williams
3            DE Rumelhart, GE Hinton, RJ Williams
4 DE Rumelhart, JL MCClelland, PDP Research Group
5            A Krizhevsky, I Sutskever, GE Hinton
6                   GE Hinton, S Osindero, YW Teh
                                                                    journal
1                                                  Nature 323, 533-536, 1986
2 Parallel Distributed Processing: Explorations in the Microstructure of ..., 1986
3                              CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR, 1985
4                                                   MIT press 1, 184, 1987
5               Advances in neural information processing systems, 1097-1105, 2012
6                                    Neural computation 18 (7), 1527-1554, 2006
  number of citations publication_year
1              34900*             1986
2              27417*             1986
3               23094             1985
4               18726             1987
5               15040             2012
6                6618             2006

> #==================Problem 2(c) =================
> #case1: if it returns HTML output
> #case2: if user input is numeric
> #case3: if user input is NA
> #Above test cases will print out the results of citation_func again but will not generate
> library(testthat)
```

```
> context("citation_func")
> test_that("citation_func handles numeric value", {
+   expect_true(is.na(citation_func("Geoffrey Hinton")) == FALSE)
+   expect_equal(citation_func(22), "input should be a string")
+   expect_equal(citation_func(NA), "input should not be NA")
+
+ })

[1] "JicYPdAAAAAJ"
[1] "https://scholar.google.com/citations?user=JicYPdAAAAAJ&hl=en&oe=ASCII&oi=ao"

> #===============Problem 2(d)==========
> #Based on observation, more contents are embedded in "<span>" and more citation results ar
> #Pass parsed html file to xpathApply function and get the xmlValue in <span>
> #r_1 <- citation_func("Geoffrey Hinton")
> r_1<-htmlParse(r_1)
> library(RCurl)
> show_more <-xpathApply(r_1,"//span[@class='gs_lbl']",xmlValue )
```