PS2

Riv Jenkins

1 a)

When the letters are saved in text format in a csv, each letter uses 1 byte, and each comma uses 1 byte, so the whole file take twice the number of letters for a total of 2 million bytes.

When the letters are smushed together by removing the commas, the file then halves back down to the number of letters (1 million). I believe the extra byte in both files is a newline.

In the binary format, the numbers are all stored right next to each other, so the only room taken up is the room necessary to store the numbers in binary.

When the numbers are saved in a csv, commas are added between each number, and each digit has to be stored instead of the whole number in binary. This greatly increases the size of the file.

When the numbers are rounded off, the number of digits stored is greatly reduced, so the size of the file correspondingly shrinks.

b)

The save function seems to perform some basic compression of the file when it saves. In the first example given, the amount of memory used decreases from 1 million to around 650 thousand. In the second example, the amount of memory decreases drastically to around 1 thousand. This is caused by the repetition of the same character in the second example. Since the same exact character is repeated 1 million times, the file only needs to store that character and the number of times it is repeated as well as some directions so that programs know how to read the file. This is what caused the reduction in size for the first example as well, but there were fewer repeated characters there since there were many different letters sampled from.

2 a)

```r
Search_Authors <- function(Name){
  #function takes a string Name and returns the html from the list of papers and the
  #author's ID for google scholar. If there is no matching author, the function returns
  #null and prints an error message. If Name is not a character string, the function
  #returns null and prints an error message.
  if(is.character(Name) == F){
    print("Please give an author name in the form of a string")
    return(NULL)
  }
  base_URL1 = "https://scholar.google.com/citations?view_op=search_authors&mauthors="
  #remove spaces in name and replace with "+"
  Name = str_replace_all(Name," ","+")
  URL = str_c(base_URL1, Name)

  #read results from search for author
  search_results = htmlParse(readLines(URL))
  Links = getHTMLLinks(search_results)
  #find all users from the search results
  mask = str_detect(Links, "user\\=")
  base_URL2 = "https://scholar.google.com"
  #contstruct the URL that leads to the author's citations page
  URL2 = str_c(base_URL2, Links[mask][1])

  #extract the ID of the author from the search results
  ID = str_extract(Links[mask][1], "user\\=[[:alpha:][0-9]]+\\&")
  ID = str_replace(ID, "user\\=", "")
  ID = str_replace(ID, "\\&", "")
  if(is.na(ID)){ #if there is no matching author, return null
    print("Could not find that author")
    return(NULL)
  }
  #output the results as a list
  Results = list("html" = readLines(URL2), "ID" = ID)
  return(Results)
}
Name = "Geoffrey Hinton"
Search_Results = Search_Authors(Name)
```

```
## Warning in readLines(URL): incomplete final line found on 'https://
## scholar.google.com/citations?view_op=search_authors&mauthors=Geoffrey
## +Hinton'

## Warning in readLines(URL2): incomplete final line found on 'https://
## scholar.google.com/citations?user=JicYPdAAAAAJ&hl=en&oe=ASCII'
```

```
print(paste("The ID of", Name, "is", Search_Results$ID))
```

```
## [1] "The ID of Geoffrey Hinton is JicYPdAAAAAJ"
```

b)

```
Papers_Table <- function(html){
  #This function reads the raw html from an author's citations page and outputs a
  #dataframe which contains the title, authors, year, journal, and number of citations
  #for the author's top 20 papers

  #Exctract the authors and journal information
  authors = str_extract_all(html, "\"gs_gray\">[^<]+</div>")
  journals = str_extract_all(html, "\"gs_gray\">[^<]+<span")
  authors = str_replace_all(unlist(authors), "\"gs_gray\">", "")
  journals = str_replace_all(unlist(journals), "\"gs_gray\">", "")
  authors = str_replace_all(authors, "</div>", "")
  journals = str_replace_all(journals, "<span", "")

  #year, cited number, and title can be read using builtin table reading function
  df = readHTMLTable(html, which = 2)

  #get rid of the extra stuff after the "Title" heading
  df$Title = df[[grep("Title.+", attributes(df)$names)]]
  df[[grep("Title.+", attributes(df)$names)]] = NULL
```

```r
  #add authors and journal columns
  df$Authors = authors
  df$Journal = journals

  return(df)
}
Papers = Papers_Table(Search_Results$html)
```

c)

```r
test_that("The author search function works correctly", {
  expect_null(Search_Authors("adlsjf lasjdf"))
  expect_null(Search_Authors(5))
  expect_length(Search_Authors("Geoffrey Hinton"), 2)
  expect_true(length(Search_Authors("Dorit Hochbaum")$html) > 0)
  expect_type(Search_Authors("Geoffrey Hinton")$ID, "character")
})
```

```
## [1] "Could not find that author"
## [1] "Please give an author name in the form of a string"
```

```r
Search_Results = Search_Authors("Geoffrey Hinton")
```

```
## Warning in readLines(URL): incomplete final line found on 'https://
## scholar.google.com/citations?view_op=search_authors&mauthors=Geoffrey
## +Hinton'

## Warning in readLines(URL2): incomplete final line found on 'https://
## scholar.google.com/citations?user=JicYPdAAAAAJ&hl=en&oe=ASCII'
```

```r
test_that("The table constructer works correctly", {
  expect_s3_class(Papers_Table(Search_Results$html), "data.frame")
  expect_true("Title" %in% names(Papers_Table(Search_Results$html)))
  expect_true("Cited by" %in% names(Papers_Table(Search_Results$html)))
  expect_true("Year" %in% names(Papers_Table(Search_Results$html)))
  expect_true("Authors" %in% names(Papers_Table(Search_Results$html)))
  expect_true("Journal" %in% names(Papers_Table(Search_Results$html)))
  expect_equal(dim(Papers_Table(Search_Results$html))[1], 20)
})
```