STAT243 - Problem Set 2

Cristobal Pais - cpaismz@berkeley.edu

September 15th, 2017

Problem 1

a) Comparing file sizes

Before starting the analysis of each file and its size, important is to note the fact that depending on the operating system (Windows or UNIX in this case) there are some differences in terms of the size of the files. In Windows the size of a new line character is 2 bytes while in UNIX it is only 1 byte. The same situation occurs with the end of the file (EOF) character. Therefore, executing the R-code (exactly the same code in both operating systems) provided in the following questions in Windows OS will modify the results obtained (size will be larger in Windows files). Thus, we perform our analysis for both operating systems, generating the files from R in UNIX -via R for UNIX using the UBUNTU installation in Windows 10- and from R in Windows, covering both operating systems cases for completeness.

i) In the case of the first file (tmp1.csv), we are including 1 million characters inside a .csv file, where (by default) will be separated by a new line character. Hence, the size of the file will be equal to 1e6 bytes (characters) plus the size of the corresponding new lines and *end of the file* characters as follows:

```r
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

## save letters in text format
chars <- sample(letters, 1e6, replace = TRUE)
write.table(chars, file = 'tmp1.csv', row.names = FALSE, quote = FALSE,
            col.names = FALSE)
```

Once we created the files (in */ps2/windows* when using the Windows version of R) we can check the size of each file and analyze it:

- UNIX: we have 1e6 characters + 1e6 new lines characters, thus, we have 1e6+1e6 = 2e6 bytes in total since each character has a size of 1 byte by definition

```r
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
        ls -l tmp1.csv
        file tmp1.csv"', intern = TRUE)
```

```
## [1] "-rw-rw-rw- 1 cpaismz cpaismz 2000000 sep 10 06:20 tmp1.csv"
## [2] "tmp1.csv: ASCII text"
```

- Windows: we have 1e6 characters + 1e6 new lines characters. However, in Windows OS a new line character has a size of 2 bytes each and thus, we have 1e6+2e6 = 3e6 bytes in total.

```
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
        ls -l tmp1.csv
        file tmp1.csv"', intern = TRUE)
```

```
## [1] "-rwxrwxrwx 1 cpaismz cpaismz 3000000 sep 10 06:00 tmp1.csv"
## [2] "tmp1.csv: ASCII text, with CRLF line terminators"
```

Based on the previous outputs, we can clearly see the consistency of the previous analysis.

ii) For the second file (tmp2.csv), we are including 1 million characters inside a .csv file without separations between characters. Hence, the size of the file will be equal to 1e6 bytes (characters) plus the size of the corresponding end of the file character:

```
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

## save letters in text format without spaces
chars <- paste(chars, collapse = '')
write.table(chars, file = 'tmp2.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE)
```

Once we created the files (in */ps2/windows* when using the Windows version of R) we can check the size of each file and analyze it:

- UNIX: we have 1e6 characters + 1 end of the file (EOF, 26 in ASCII) character, thus, we have 1e6+1 = 1.000001e6 bytes in total since each character has a size of 1 byte by definition.

```
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
        ls -l tmp2.csv
        file tmp2.csv"', intern = TRUE)
```

```
## [1] "-rw-rw-rw- 1 cpaismz cpaismz 1000001 sep 10 06:22 tmp2.csv"
## [2] "tmp2.csv: ASCII text, with very long lines, with no line terminators"
```

- Windows: we have 1e6 characters + 1 end of the file character. However, in Windows an end of the file (EOF) character has a size of 2 bytes each and thus, we have 1e6+2 = 1.000002e6 bytes in total.

```
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
        ls -l tmp2.csv
        file tmp2.csv"', intern = TRUE)
```

```
## [1] "-rwxrwxrwx 1 cpaismz cpaismz 1000002 sep 10 06:00 tmp2.csv"
## [2] "tmp2.csv: ASCII text, with very long lines, with no line terminators"
```

iii) For the third file (tmp3.csv), we are including 1 million of random numbers (taking from a normal distribution) inside a .csv file. As a general analysis for the rest of the files, it is important to note that since we are creating a random vector of numbers following a standard normal distribution

$N(0,1)$, besides the number of decimals that we are including (random, depends on the number generated), we have to take into account the number of negative numbers (extra character '-' at the beginning) and positive numbers. In addition, we have the new line characters at the end of each line.

Furthermore, we have to take into account the fact that the file format is a .Rda file generated by the save function. This function can take any object in R and represent it in a binary format that can be read by a computer but it is not human readable. R binary files extension .Rda are used to keep R object "AS-IS" with the benefit of a reduction in the size of the file due to its ability to reduce the file size, even more (as we will see with the file tmp7.csv) when some character patterns can be identified.

As the documentation of the function indicates, "to maximize the number of observations in a data package, the data needs to be compressed. Compression is done through an algorithm that creates a dictionary table of more commonly used patterns. This enables the reduction of the file size as large patterns are reduced to smaller patterns". There are three different types of compression offered by the function: *gzip*, *bzip2*, and *xz*. By default, the *save()* function will perform a *gzip* compression to the .Rda files, a compression format that reduces the size of the given files using the well-known Lempel-Ziv coding (LZ77).

Based on the previous analysis, we are aware of the fact that the file will be non-human readable, compressed using *gzip* and therefore, we will not have a match between the number of characters and the size of the file:

```r
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

## save in binary format
nums <- rnorm(1e6)
save(nums, file = 'tmp3.Rda')
```

Once we created the files (in */ps2/windows* when using the Windows version of R) we can check the size of each file and analyze it:

- UNIX: we do not know the total number of characters, hence, we can count them using the *wc -m* command and we can then check the size of the file and its format in order to reflect the facts stated by our previous analysis.

```r
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       wc -m tmp3.Rda
       ls -l tmp3.Rda
       file tmp3.Rda"', intern = TRUE)
```

```
## [1] "3974998 tmp3.Rda"
## [2] "-rw-rw-rw- 1 cpaismz cpaismz 7678196 sep 10 06:22 tmp3.Rda"
## [3] "tmp3.Rda: gzip compressed data, from Unix"
```

- Windows: as in the UNIX case, we count the number of characters and check the file's size and format in order to check our general analysis.

```
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
       wc -m tmp3.Rda
       ls -l tmp3.Rda
       file tmp3.Rda"', intern = TRUE)
```

```
## [1] "3977053 tmp3.Rda"
## [2] "-rwxrwxrwx 1 cpaismz cpaismz 7678064 sep 10 06:00 tmp3.Rda"
## [3] "tmp3.Rda: gzip compressed data, from HPFS filesystem (OS/2, NT)"
```

Based on the outputs from these codes, we can easily check that we no longer have a perfect relationship between the number of characters inside the file and the size (they are not equal). In addition, there is still a difference between UNIX and Windows file's size. Important is to note that the *file* command indicates in both cases that we are dealing with a gzip file. Hence, there are no inconsistencies with these values (sizes). **Note**: UNIX and Windows files for tmp3.csv were not created using the same random seed and therefore, their sizes do not need to match or be similar.

iv) For the fourth file (tmp4.csv), we are including the same 1 million random numbers but in this case, we are saving it in a .csv file using the *write.table()* function in R. Thus, no compression is performed and we can follow the same analysis from the beginning of this section: we can count the number of characters inside the file, such that we know that one character represents a size of 1 byte in the .csv file (numbers, negative signs and new line characters).

```
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

## save in text format
write.table(nums, file = 'tmp4.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE, sep = ',')
```

Once we created the files (in */ps2/windows* when using the Windows version of R) we can check the size of each file and analyze it:

- UNIX: we do not know the total number of characters, hence, we can count them using the *wc -m* command and we can then check the size of the file and its format.

```
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       wc -m tmp4.csv
       ls -l tmp4.csv
       file tmp4.csv"', intern = TRUE)
```

```
## [1] "18152709 tmp4.csv"
## [2] "-rw-rw-rw- 1 cpaismz cpaismz 18152709 sep 10 06:22 tmp4.csv"
## [3] "tmp4.csv: ASCII text"
```

- Windows: same as with UNIX, we do not know the total number of characters, hence, we can count them using the *wc -m* command and we can then check the size of the file and its format.

```
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
       wc -m tmp4.csv
       ls -l tmp4.csv
       file tmp4.csv"', intern = TRUE)
```

```
## [1] "19159795 tmp4.csv"
## [2] "-rwxrwxrwx 1 cpaismz cpaismz 19159795 sep 10 06:00 tmp4.csv"
## [3] "tmp4.csv: ASCII text, with CRLF line terminators"
```

Based on the previous outputs, we can clearly check that the size of the files is equal to the number of characters inside them. Hence, our analysis is consistent with the fact that no compression has been made to any of these files. **Note**: UNIX and Windows files for tmp4.csv were not created using the same random seed and therefore, their sizes do not need to match or be similar.

v) For the fifth file (tmp5.csv), we are including the same 1 million random numbers but in this case, we are rounding them to (up to: for example 0 remains as 0 and 0.X are rounded down to 0.) two decimals before saving them in to .csv file using the *write.table()* function in R. Again, no compression is performed and we can follow the same analysis from the previous section: we can count the number of characters inside the file, such that we know that one character represents a size of 1 byte in the .csv file (numbers, negative sign, and new line characters).

   In addition, we can easily check that the new size of the file is smaller than in tmp4.csv:

```
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

## save in text format with rounded numbers (up to 2 decimals)
write.table(round(nums, 2), file = 'tmp5.csv', row.names = FALSE,
quote = FALSE, col.names = FALSE, sep = ',')
```

Once we created the files (in ~/ps2/windows when using the Windows version of R) we can check the size of each file and analyze it:

- UNIX: we do not know the total number of characters, hence, we can count them using the *wc -m* command and we can then check the size of the file and its format.

```
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       wc -m tmp5.csv
       ls -l tmp5.csv
       file tmp5.csv"', intern = TRUE)
```

```
## [1] "5377265 tmp5.csv"
## [2] "-rw-rw-rw- 1 cpaismz cpaismz 5377265 sep 10 06:22 tmp5.csv"
## [3] "tmp5.csv: ASCII text"
```

- Windows: we do not know the total number of characters, hence, we can count them using the *wc -m* command and we can then check the size of the file and its format.

```r
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
       wc -m tmp5.csv
       ls -l tmp5.csv
       file tmp5.csv"', intern = TRUE)
```

```
## [1] "6378819 tmp5.csv"
## [2] "-rwxrwxrwx 1 cpaismz cpaismz 6378819 sep 10 06:00 tmp5.csv"
## [3] "tmp5.csv: ASCII text, with CRLF line terminators"
```

Based on the previous outputs, we can clearly check that the size of the files is equal to the number of characters inside them. Hence, our analysis is consistent with the fact that no compression has been made to any of these files. Note: UNIX and Windows files for tmp5.csv were not created using the same random seed and therefore, their sizes do not need to match or be similar.

b) Compression and repetitions

As we already indicated in the previous section, the *save()* function performs a *gzip* compression by default, creating a binary file *.rda* that follows that compression method. Therefore, we end up with a non-human readable file with a smaller size due to the compression process. In particular, the *gzip* compression method uses the Lempel-Ziv coding (LZ77), an algorithm that compresses the file by replacing repeated occurrences inside the data file using references to a single copy of that particular data in the uncompressed data stream. As we can find in [1], "*rather than use an abbreviation table, LZ77 cleverly leaves the first occurrence of a repeated wor d in its original position in the text, and when the repetition occurs, the repeated word is replaced by: go backwards XX characters and copy the YY letters at that point*".

This overall process can be understood inside the "data deduplication'‘concept, a technique for eliminating duplicate copies of repeating data. In [2], this process is defined as a process where " *unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced*".

We will analyze the different sizes of the file tmp6.Rda and tmp7.Rda using the previous definitions and information.

```r
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

chars <- sample(letters, 1e6, replace = TRUE)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp6.Rda')
```

Once we created the files (in */ps2/windows* when using the Windows version of R) we can check the size of each file and analyze it:

```r
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       wc -m tmp6.Rda
       ls -l tmp6.Rda
       file tmp6.Rda"', intern = TRUE)
```

```
## [1] "344641 tmp6.Rda"
## [2] "-rw-rw-rw- 1 cpaismz cpaismz 635323 sep 12 04:21 tmp6.Rda"
## [3] "tmp6.Rda: gzip compressed data, from Unix"
```

- UNIX: we know the total number of characters, however, since the generated file is a compressed binary one -and as we already mentioned in the previous section- we cannot use the number of characters as the size of the file.

```
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
       wc -m tmp6.Rda
       ls -l tmp6.Rda
       file tmp6.Rda"', intern = TRUE)
```

```
## [1] "344457 tmp6.Rda"
## [2] "-rwxrwxrwx 1 cpaismz cpaismz 635217 sep 14 01:53 tmp6.Rda"
## [3] "tmp6.Rda: gzip compressed data, from HPFS filesystem (OS/2, NT)"
```

- Windows: same as for UNIX.

In order to check and complete our analysis, we will generate four new files using different options for the save command:

1. The first one will use the ascii flag, generating a non-compressed file
2. The second one will be an ascii file that is compressed using *gzip* format
3. The third one will consist of a non-ascii file without compression.
4. The last one will be a non-ascii file with *xz* compression

```
# Generate the extra files for the analysis (different save options)
save(chars, file = 'tmp6_V2.Rda', ascii = TRUE)
save(chars, file = 'tmp6_V3.Rda', ascii = TRUE, compress = "gzip")
save(chars, file = 'tmp6_V4.Rda', ascii = FALSE, compress = FALSE)
save(chars, file = 'tmp6_V5.Rda', ascii = FALSE, compress = "xz")
save(chars, file = 'tmp6_V6.Rda', ascii = FALSE, compress = "gzip")
```

- UNIX and Windows: Now we can compare all files sizes and types. Clearly, we expect smaller file sizes for those where a compression is performed (Original, V3, V5 and V6) while for the others (V2 and V4), we would be able to obtain their sizes (larger sizes) by simply count the number of characters inside them (no compression has been made).

```
# Check files sizes and types by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       echo \\"Files sizes comparison:\\"
       ls -l tmp6.Rda tmp6_V2.Rda tmp6_V3.Rda tmp6_V4.Rda tmp6_V5.Rda tmp6_V6.Rda
       echo \\"Files types comparison:\\"
       file tmp6.Rda tmp6_V2.Rda tmp6_V3.Rda tmp6_V4.Rda tmp6_V5.Rda tmp6_V6.Rda"',
       intern = TRUE)
```

```
##  [1] "Files sizes comparison:"
##  [2] "-rw-rw-rw- 1 cpaismz cpaismz  635323 sep 12 04:21 tmp6.Rda"
##  [3] "-rw-rw-rw- 1 cpaismz cpaismz 1000070 sep 12 04:21 tmp6_V2.Rda"
##  [4] "-rw-rw-rw- 1 cpaismz cpaismz  635340 sep 12 04:21 tmp6_V3.Rda"
##  [5] "-rw-rw-rw- 1 cpaismz cpaismz 1000060 sep 12 04:21 tmp6_V4.Rda"
##  [6] "-rw-rw-rw- 1 cpaismz cpaismz  606296 sep 12 04:22 tmp6_V5.Rda"
##  [7] "-rw-rw-rw- 1 cpaismz cpaismz  635323 sep 12 04:22 tmp6_V6.Rda"
##  [8] "Files types comparison:"
##  [9] "tmp6.Rda:    gzip compressed data, from Unix"
## [10] "tmp6_V2.Rda: ASCII text, with very long lines"
## [11] "tmp6_V3.Rda: gzip compressed data, from Unix"
## [12] "tmp6_V4.Rda: data"
## [13] "tmp6_V5.Rda: XZ compressed data"
## [14] "tmp6_V6.Rda: gzip compressed data, from Unix"
```

As we expected based on the *save()* function arguments, the original file and the V6 file have exactly the same size (and they are the same type) because we are simply given the default options (explicitly) for generating the sixth version of the file. On the other hand, we can see that V2 and V4 reached the largest sizes due to the lack of compression. Interesting is to note that the *xz* compression obtains the best performance in terms of reducing ratio (smallest file size). Note also that V2 and V4 differ in their format: while V2 is treated as a very long ASCII text, V4 is identified as a data file, therefore, some (small) differences in terms of sizes are detected.

Now, we can generate the file with repeated characters (tmp7.Rda) in order to compare its size with tmp6.Rda and check our initial analysis regarding the performance of the compression algorithms were repeated patterns can be found inside the data:

```r
# Setting the working directory (UNIX)
setwd("~/ps2/unix")

# Create the file with repeated characters
chars <- rep('a', 1e6)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp7.Rda')
```

Once we created the files (in */ps2/windows* when using the Windows version of R) we can check the size of each file and analyze it:

```r
# Check file size and type by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       wc -m tmp7.Rda
       ls -l tmp7.Rda
       file tmp7.Rda"', intern = TRUE)
```

```
## [1] "1020 tmp7.Rda"
## [2] "-rw-rw-rw- 1 cpaismz cpaismz 1056 sep 12 04:11 tmp7.Rda"
## [3] "tmp7.Rda: gzip compressed data, from Unix"
```

- UNIX: as before, we know the total number of characters, however, since the generated file is a compressed binary one -and as we already mentioned in the previous section- we cannot use the number of characters as the size of the file.

```r
# Check file size and type by calling bash from windows: Windows format
system('bash -c "cd ~/ps2/windows
       wc -m tmp7.Rda
       ls -l tmp7.Rda
       file tmp7.Rda"', intern = TRUE)
```

```
## [1] "1020 tmp7.Rda"
## [2] "-rwxrwxrwx 1 cpaismz cpaismz 1056 sep 14 01:53 tmp7.Rda"
## [3] "tmp7.Rda: gzip compressed data, from HPFS filesystem (OS/2, NT)"
```

- Windows: same as for UNIX.

Again, we generate the extra files using different arguments inside the *save()* function for completeness:

```r
save(chars, file = 'tmp7_V2.Rda', ascii = TRUE)
save(chars, file = 'tmp7_V3.Rda', ascii = TRUE, compress = "gzip")
save(chars, file = 'tmp7_V4.Rda', ascii = FALSE, compress = FALSE)
save(chars, file = 'tmp7_V5.Rda', ascii = FALSE, compress = "xz")
save(chars, file = 'tmp7_V6.Rda', ascii = FALSE, compress = "gzip")
```

- UNIX and Windows: Now we can compare all files sizes and types. Clearly, we expect smaller file sizes for those where a compression is performed (Original, V3, V5 and V6) while for the others (V2 and V4), we would be able to obtain their sizes (larger sizes) by simply count the number of characters inside them (no compression has been made).

```r
# Check files sizes and types by calling bash from windows: UNIX format
system('bash -c "cd ~/ps2/unix
       echo \\"Files sizes comparison:\\"
       ls -l tmp7.Rda tmp7_V2.Rda tmp7_V3.Rda tmp7_V4.Rda tmp7_V5.Rda tmp7_V6.Rda
       echo \\"Files types comparison:\\"
       file tmp7.Rda tmp7_V2.Rda tmp7_V3.Rda tmp7_V4.Rda tmp7_V5.Rda tmp7_V6.Rda"',
       intern = TRUE)
```

```
##  [1] "Files sizes comparison:"
##  [2] "-rw-rw-rw- 1 cpaismz cpaismz    1056 sep 12 04:11 tmp7.Rda"
##  [3] "-rw-rw-rw- 1 cpaismz cpaismz 1000070 sep 12 04:11 tmp7_V2.Rda"
##  [4] "-rw-rw-rw- 1 cpaismz cpaismz    1057 sep 12 04:11 tmp7_V3.Rda"
##  [5] "-rw-rw-rw- 1 cpaismz cpaismz 1000060 sep 12 04:11 tmp7_V4.Rda"
##  [6] "-rw-rw-rw- 1 cpaismz cpaismz     320 sep 12 04:11 tmp7_V5.Rda"
##  [7] "-rw-rw-rw- 1 cpaismz cpaismz    1056 sep 12 04:42 tmp7_V6.Rda"
##  [8] "Files types comparison:"
##  [9] "tmp7.Rda:    gzip compressed data, from Unix"
## [10] "tmp7_V2.Rda: ASCII text, with very long lines"
## [11] "tmp7_V3.Rda: gzip compressed data, from Unix"
## [12] "tmp7_V4.Rda: data"
## [13] "tmp7_V5.Rda: XZ compressed data"
## [14] "tmp7_V6.Rda: gzip compressed data, from Unix"
```

As expected, the sizes' pattern is the same as with the previous file, however, file sizes are clearly smaller than the ones obtained with the random vector inside tmp6.Rda, although the number of characters remains the same. The explanation behind this pattern lies on the previous results and the general analysis stated at the beginning of this section, the file tmp7.Rda presents a reduced size in comparison to tmp6.csv due to the fact that it contains only one character that is repeated 1 million times. Hence, the compression algorithm can take advantage of it using the fact that repetitions can be referenced in order to reduce the size of the file, obtaining very good compression ratios. Since we know the way that *gzip* works, we can easily understand that the difference in sizes radicates in this deduplication process performed by the compression.

Finally, note that as we expected, the size of tmp6_V2.Rda and tmp4_V4.Rda files are equal to the ones obtained in tmp7_V2.Rda and tmp7_V4.Rda respectively. The explanation is simple: in both cases no compression is performed and thus the size of the files is simply the total number of characters inside them and there is no difference between recording a random vector of characters or one that consists of a unique (repeated) character.

Problem 2

The general strategy to solve this problem consists of gathering all the necessary information from the *.html* files that can be found in the relevant Google Schoolar urls. At the beginning, our code obtains the *.html* file that defines the website reached right after entering the name of the researcher in http://scholar.google.com. Based on the structure of this file and the pattern of the queries, we can obtain the user ID (if the name is a valid and registered author's name) and the corresponding link to its personal citation website. Using this information, we can obtain the HTML file associated with the author's personal citation website. All these operations are performed by our first function called *CitesScholar()* such that the input is the name of the researcher and the output is a list object containing both the user ID and the HTML file from the personal citation website.

Once we have a parsed HTML object from the author's personal website, we can easily analyze it and check/understand its structure. Thus, we are able to perform a series of queries using the XML library functions and some post-processing with the help of the stringr library. By the end of the process, we obtain five vectors with all the relevant information per article: Title, Authors, Journal, Year, and number of Citations. Note that some authors may have some articles where not all the information is available (blank fields) or papers that appear multiple times with different number of citations, in those cases, we leave them 'as-is' in our table, being consistent with the information provided by the Google Scholar website.

Since Google Scholar website could try to block our connections after repeating the procedure, we develop an extra function called *DownloadfromScholar()* that, instead of creating a HTML file on the fly using *readLines()* function, downloads the relevant *.html* files to the current working directory and then parses them. Thanks to this implementation, we are able to reuse previous queries and create their correspondent DataFrames. Thus, we can use this function as an alternative when our original implementation is being blocked.

All the previous operations, starting from the parsed HTML file (from the personal citation website) and ending with a DataFrame with all the relevant information are performed by our function *CreateDataFrame_subset()* when dealing with only the first 20 publications and by our function *CreateDataFrame()* for the case when we include all the author's publications inside our DataFrame. A slight modification to the original link and a loop must be added in order to be able to perform a series of queries that allow us to get all the publications.

Finally, some tests are performed to our functions using the package testthat as required in section c).

Thus, we have the following steps:

i) Download/extract the *.html* file associated with the provided author's name.

ii) Obtain the user ID and personal website link from it.

iii) Download/extract the HTML file from the author's profile, parse it.

iv) Process it and create the relevant columns.

v) Return a DataFrame containing the relevant information.

a) Google Scholar citation function: ID and html

The function's input consists of the author's name, only allowing names that contain a first and a last name separated by a blank space, following the example provided for Geoffrey Hinton. Its outputs are the user ID and the parsed HTML associated with the author's personal citation website.

The implementation is as follows:

1. We declare our function *CitesScholar()* that requires the name of the author as main input. In order to get access to all the functions that are used inside of it, the main libraries are invoked and warnings are suppresed for visualization purposes. In addition, two "valid input" checks are performed: (1)

determining if the input is a character or not. If not, then an error message is printed to the main console and the function is no longer executed, and (2) checking if the given name follows the specific pattern we want (using a regular expression) from the user where a first name must be followed by an empty space and then by the last name of the author. If not, an error message with a usage example is given to the user via the console terminal.

```r
# Function declaration
CitesScholar <- function(name) {
  # Loading libraries
  library(XML)
  library(RCurl)
  library(stringr)

  # Turning off the warnings for visualization purposes
  options(warn = -1)

  # Check if the input is a string
  if (!is.character(name)){
    stop("The input must be a name (e.g Geoffrey Hinton)")
  }

  # Check if we have only letters and a first and last name
  if (grepl("^[[:alpha:]]+[[:space:]]+[[:alpha:]]+$", name) == FALSE){
    stop(paste("Only two words (no digits) are allowed as the author's name.",
               "The name string must have a space between the first and last names,",
               "please try again (e.g Geoffrey Hinton)", sep = " "))
  }
```

2. After passing the checks, we know that the user has given a valid input, thus, we extract the first and last names from the original variable name based on the blank space between them and the initial url is generated based on the structure followed by the Google Scholar website. This url is the one that a normal user will get after writing the name of the author in the search field of the Google Scholar main web (http://scholar.google.com)

```r
  # Separate the name in first and last names
  firstname <- strsplit(name, " ")[[1]][[1]]
  lastname <- strsplit(name, " ")[[1]][[2]]

  # Generate the URL with the profile in google scholar
  urlInit <- paste("https://scholar.google.com/scholar?hl=es&q=", firstname,
                   "+", lastname, "&btnG=&lr=", sep = "")
```

3. We extract the HTML file information from the generated url using the classic *readLines()* function, obtaining a rawHTML file. Then we parse this file thanks to the function included in the XML library called *htmlParse()*. Once we have the parsed HTML object, we can obtain the link associated with the author's personal website thanks to our knowledge of the *.html* structure. Hence, we perform a query that returns the link (attribute) associated with a node that contains it.

```r
  # Get the Google Scholar ID and the link to the personal citations website
  rawHTML <- readLines(urlInit)

  # Parse the raw HTML code
  parsedHTML <- htmlParse(rawHTML)

  # Find the link to the author's personal website
  link <- xpathSApply(parsedHTML, "//h4[@class = 'gs_rt2']//a",
                      xmlGetAttr, "href")
```

4. We include an additional check that validates the existence of a link. If the query returns an empty value/string, it means that the author does not have a personal citation website and therefore, it does not exist inside the Google Scholar Database as a registered researcher. In that case, an error message is given to the user.

```
  # Check if there are valid links inside the file: if not, stop
  if (length(link) <= 0) {
    stop(paste("The author does not exist in the Google Scholar Database,",
               "please try again with a different name", sep = " "))
  }
```

5. Once the link to the personal website is obtained, it can be processed using regular expressions in order to extract the user ID from it as a string. Again, an extra safety check is included, testing the case that no user ID is found indicating that there is no personal website available for the given author name.

```
  # Extract the user
  UserID <- str_match(link, ".*user=([\\d\\w]+)&")[[2]]

  # If no User ID, then author does not exists in Google Scholar: stop
  if (is.na(UserID) == TRUE){
    stop(paste("The author does not exist in the Google Scholar Database,",
               "please try again with a different name", sep = " "))
  }
```

6. Finally, the url associated with the author's personal citation website is generated based on the structure that we previously identified from the Google Scholar website. A new HTML extraction is performed via the *readLines()*. After parsing the resulting object, a list containing both the user ID and parsed HTML code is created in order to be able to return multiple objects via the return statament. This is a peculiarity of R: the return statement does not allow multiple arguments, hence, we can create a list that contains all the elements that we want to return from the function.

```
  # Get the citations html
  urlcites <- paste("https://scholar.google.com", link, sep = "")
  htmlcites <- readLines(urlcites)
  parhtmlcites <- htmlParse(htmlcites)

  # Return the user ID and parsed html from the personal citations web
  returnlist <- list(UserID, parhtmlcites)
  return(returnlist)
}
```

Therefore, the function returns the user ID and the HTML object (parsed) that contains all the information of the author's personal website. From this, we will be able to extract the title, journal, authors, year of publication, and number of citations of each of his/her published articles.

a.2) Extra function

As we indicated above, we developed a second function for performing the previous operations in order to have an alternative method in the case that Google Scholar blocks our queries. In this case, instead of extracting the HTML file information using the *readLines()* function, we are downloading the relevant *.html* files and then we are processing them via the *download.file()* function (similar to the wget function in bash).

Our alternative function is as follows:

1. All the code (except for the name of the function) is identical up to the second chunk of code stated above. Then, for the third chunk of code, instead of using the *readLines()* function, we download the content of the html file into a *.html* file with the name of the author. After downloading it, we simply parse its content into a new variable.

```
  # Generate the initial URL to download the .html file
  urlInit <- paste("https://scholar.google.com/scholar?q=", firstname,
                   "+", lastname, "&hl=en&as_sdt=0,5", sep = "")
  filename <- paste(firstname, lastname, ".html", sep = "")

  # Download the file and parse the html file
  download.file(urlInit, filename)
  parsedHTML <- htmlParse(readLines(filename))
```

2. Then, the other modification occurs at the end of the function (chunk number 6 in the previous function) where we again download the HTML file from the author's personal citation website instead of extracting it using the *readLines()* function. We parse it and we create the list for returning both the user ID and the parsed HTML object.

```
  # Get the citations html: generate the url and download it
  urlcites <- paste("https://scholar.google.com", link, sep = "")
  filenamecites <- paste(UserID, ".html", sep = "")
  download.file(urlcites, filenamecites)

  # Parse the new html
  parhtmlcites <- htmlParse(readLines(filenamecites))

  # Return the user ID and parsed html from the personal citations web
  returnlist <- list(UserID, parhtmlcites)
  return(returnlist)
}
```

With this slight modifications, we are able to perform queries when our previous function is being blocked by the Google Scholar server. We can easily use the *CitesScholar()* (or the *DownloadfromScholar()*) function as follows:

```
# Obtain the User ID and HTML object of the author
ResultsList <- CitesScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Display html class
class(Authorhtml)[[1]]
```

```
## [1] "HTMLInternalDocument"
```

```
# Display outputs (summary of the html file for visualization purposes)
# User ID
UserID
```

```
## [1] "JicYPdAAAAAJ"
```

```
# HTML summary
summary(Authorhtml)
```

```
## $nameCounts
##
##   span      a    div     td     tr  input     li script button     th
##    117     97     97     69     26     15     14     12     11      9
##   form  style   meta     h2  label      p  table   body     h3   head
##      6      6      5      3      2      2      2      1      1      1
##   html    img   link  tbody  thead  title     ul
##      1      1      1      1      1      1      1
##
## $numNodes
## [1] 503
```

```
# Obtain the User ID and HTML object of the author
ResultsList <- DownloadfromScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Display html class
class(Authorhtml)[[1]]
```

```
## [1] "HTMLInternalDocument"
```

```
# Display outputs (summary of the html file for visualization purposes)
# User ID
UserID
```

```
## [1] "JicYPdAAAAAJ"
```

```
# HTML summary
summary(Authorhtml)
```

```
## $nameCounts
##
##     span        a      div       td       tr    input       li   script
##      117       97       97       69       26       15       14       13
##   button       th     form    style     meta       h2      img    label
##       11        9        6        6        5        3        2        2
##        p    table     body       h3     head     html     link noscript
##        2        2        1        1        1        1        1        1
##    tbody    thead    title       ul
##        1        1        1        1
##
## $numNodes
## [1] 506
```

b) First 20 citations as DataFrame

The following function *CreateDataFrame_subset()* takes as input the parsed HTML object obtained as the second output from the previous *CitesScholar()* function. Based on the structure of this file, a series of queries are performed in order to obtain the Title, Authors, Journals, Years and number of Citations associated with each paper/article published and indexed by the Google Scholar engine for the corresponding author. After all queries are done, some final checks regarding the length and type of vectors are performed in order to avoid errors related to missing values or wrong type entries (like strings inside the number of citations or year). Finally, a DataFrame is generated and returned to the user including the five main fields for the first 20 articles (ordered by the number of citations) presented in the author's personal citation website.

The details are as follow:

1. We declare the function including the parsed HTML object as the main output and all the necessary libraries are invoked. Warnings are omitted for visualization purposes as before.

```
# Function definition
CreateDataFrame_subset <- function(html){
  # Libraries
  library(XML)
  library(RCurl)
  library(stringr)

  # Turning off the warnings for visualization purposes
  options(warn = -1)
```

2. A first check related to the input type is performed in order to validate its format. If the type is not the adequate for the function (parsed HTML) then an error message is output to the user and the function stops. Otherwise, the first query is performed, generating the Titles vector containing the first 20 publications of the author (ordered by the number of citations, from higher to lower). A second check is performed: if the query is empty, it means that the current HTML object does not contain citations from the Google Scholar website, indicating that the author's citation page provided does not have any article or that it is not registered in Google Scholar. In this case, the function stops and an error message is prompted to the user.

```
  # Check if the file is a parsed HTML
  if (typeof(html) != "externalptr"){
    stop("Not a valid html file from author's citation website")
  }

  if (class(html)[[1]] != "HTMLInternalDocument") {
    stop(paste("Not a valid html file from author's citation website",
               "please check the format of the input file", sep = " "))
  }

  # Articles titles
  Titles <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//tr[@class ='gsc_a_tr']
                        //td[@class = 'gsc_a_t']//a[@class = 'gsc_a_at']",
                        xmlValue, trim = TRUE, encoding = "UTF-8")

  # Check if there are valid titles: if not, not a valid file
  if (length(Titles) <= 0){
    stop(paste("There is no author's citation page inside Google Scholar",
```

```r
              "associated with the html file being processed, please",
              "try again with a different file", sep = " "))
}
```

3. If all preliminary checks are passed, a query is performed in order to get the Journal and the Authors of each entry. Note that we are explicitly encoding the result of the query using UTF-8 by default, allowing the function to deal with non-common characters like European accents/characters and Slavic letters. In addition, we can notice that the query is based on the knowledge of the HTML tree structure. Once the query is performed, the journal and authors information is split using our knowledge of the resulting string from the query (Journals are located on even entries while Authors are located on odd entries). The year of publication is deleted from the journal information.

```r
# Journals and authors array
JournalAuthors <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                              tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_t']
                              //div[@class = 'gs_gray']", xmlValue,
                              trim = TRUE, encoding = "UTF-8")

# Getting the Journals (even numbers of previous vector)
Journals <- JournalAuthors[seq(2, length(JournalAuthors), 2)]

# We delete the year of publication after the Journal
Journals <- substring(Journals, 0, nchar(Journals) - 6)

# Getting the authors (odd numbers)
Authors <- JournalAuthors[seq(1, length(JournalAuthors), 2)]
```

4. More queries are performed in order to obtain the Year of publication and the number of Citations associated with each entry/paper. In order to deal with non-numerical elements (like *, empty spaces, etc.) inside the Citations vector, we transform it into a numeric vector, obtaining NA values in those conflicting entries. Then, we simply remove all the NA values from inside. Note that since Google Scholar orders the articles by number of citations, this procedure will not produce any inconsistency since in the case that some entries are null (0 citations), we will fill the Citations vector with 0s in the next step.

```r
# Years of publication
Year <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//tr[@class ='gsc_a_tr']
                    //td[@class = 'gsc_a_y']//span[@class = 'gsc_a_h']",
                    xmlValue, trim = TRUE, encoding = "UTF-8")

# Number of citations (including repeated articles)
Citations <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']
                         //tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_c']
                         //a", xmlValue, trim = TRUE, encoding = "UTF-8")

# Delete potential non numerical elements by transforming and filtering the array
Citations <- as.numeric(Citations)
remove <- c(NA)

# Re-Transform it to a character array for the DF
Citations <- as.character(Citations[!Citations %in% remove])
```

5. If there are some articles without Citations, we fill those entries with 0s in order to match the length of the vector with the other fields for creating a consistent DataFrame.

```
  # Dimension checks: fill with 0 empty citations
  if (length(Citations) != length(Titles)){
    Citations <- c(Citations, integer(length(Titles) - length(Citations)))
  }
```

6. Finally, we declare the name of each column of the DataFrame and we generate it from the data gathered by the previous queries.

```
  # Create a DataFrame
  x_name <- "Title"
  y_name <- "Authors"
  z_name <- "Journals"
  w_name <- "Year"
  v_name <- "Citations"

  # Create the DF using the reshape2 melt function for simplicity
  require(reshape2)
  ScholarDF <- melt(data.frame(Titles, Authors, Journals, Year, Citations))

  # Define the columns' names
  colnames(ScholarDF) <- c(x_name, y_name, z_name, w_name, v_name)

  # Return the DataFrame
  return(ScholarDF)
}
```

Therefore, the function returns the DataFrame with the first 20 publications of the author under study based on the parsed HTML code obtained from the previous function *CitesScholar()* (or *DownloadfromScholar()*).

We can easily use one of these functions and then we can run our *CreateDataFrame_subset()* function:

```
# Call the function and create a DataFrame from the author's website
ResultsList <- CitesScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Create the data frame
SubsetScholarDF <- CreateDataFrame_subset(Authorhtml)

# Check the length of any column
length(SubsetScholarDF$Title)
```

```
## [1] 20
```

```
# Printing a summary using the apply function (``pretty'' output)
apply(SubsetScholarDF, 2, function(x)
  if (length(x) > 15 && is.na(str_match(x, "^[[:digit:]]"))) {
    paste(substr(x, start = 0, stop = 15), "...", sep = "")
  }
  else {substr(x, start = 0, stop = length(x))
  }
)
```

```
##       Title                Authors              Journals
##  [1,] "Learning repres..." "DE Rumelhart, G..." "Nature 323, 533..."
##  [2,] "Learning intern..." "DE Rumelhart, G..." "Parallel Distri..."
##  [3,] "Learning intern..." "DE Rumelhart, G..." "CALIFORNIA UNIV..."
##  [4,] "Parallel distri..." "DE Rumelhart, J..." "MIT press 1, 18..."
##  [5,] "Imagenet classi..." "A Krizhevsky, I..." "Advances in neu..."
##  [6,] "A fast learning..." "GE Hinton, S Os..." "Neural computat..."
##  [7,] "Parallel distri..." "JL McClelland, ..." "MIT press..."
##  [8,] "Reducing the di..." "GE Hinton, RR S..." "science 313 (57..."
##  [9,] "Deep learning..."   "Y LeCun, Y Beng..." "Nature 521 (755..."
## [10,] "Adaptive mixtur..." "RA Jacobs, MI J..." "Neural computat..."
## [11,] "Dropout: a simp..." "N Srivastava, G..." "Journal of mach..."
## [12,] "A learning algo..." "DH Ackley, GE H..." "Cognitive scien..."
## [13,] "Visualizing dat..." "L van der Maate..." "Journal of Mach..."
## [14,] "Deep neural net..." "G Hinton, L Den..." "IEEE Signal Pro..."
## [15,] "Training produc..." "GE Hinton..."       "Neural computat..."
## [16,] "A view of the E..." "RM Neal, GE Hin..." "Learning in gra..."
## [17,] "Phoneme recogni..." "A Waibel, T Han..." "IEEE transactio..."
## [18,] "Improving neura..." "GE Hinton, N Sr..." "arXiv preprint ..."
## [19,] "Rectified linea..." "V Nair, GE Hint..." "Proceedings of ..."
## [20,] "Connectionist l..." "GE Hinton..."       "Artificial inte..."
##       Year   Citations
##  [1,] "1986" "34900"
##  [2,] "1986" "27417"
##  [3,] "1985" "23094"
##  [4,] "1987" "18726"
##  [5,] "2012" "15040"
##  [6,] "2006" "6618"
##  [7,] "1987" "6477"
##  [8,] "2006" "5614"
##  [9,] "2015" "3793"
## [10,] "1991" "3551"
## [11,] "2014" "3515"
## [12,] "1985" "3316"
## [13,] "2008" "3149"
## [14,] "2012" "3147"
## [15,] "2002" "2949"
## [16,] "1998" "2468"
## [17,] "1989" "2396"
## [18,] "2012" "2111"
## [19,] "2010" "2040"
## [20,] "1989" "1814"
```

c) Testing our code

i) We have already included checks inside all our functions, hence, we develop some tests using the *testthat* library in the next section.

ii) In order to perform some tests using the *testthat* library in our two functions *CitesScholar()* and *CreateDataFrame_subset()*, we generate two *.R* files containing them and two associated test files: *CitesScholar.R*, *CreateDataFrame_subset()*, and *test_CitesScholar.R*, *test_CreateDataFrame_subset()* respectively.

CitesScholar Tests

For this function we perform three main types of tests: (1) Test the function output using a known researcher name as input, (2) Test the function using a non-registered Google Scholar author and (3) Check different input possibilities and how our function is handling them.

1. We check if the output obtained is a list if its length is valid (2) including the user ID + HTML object, and we check the type of each component inside the list. We test it with a well-known researcher name such as Albert Einstein.

```r
# Loading the function file to be tested
source("CitesScholar.R")

# Testing the CitesScholar function (input = HTML, output = list)
test_that("Types and lengths are consistent when using a real author's name", {

  # As an example, we are using the citation website of the well known Albert Einstein
  ResultsList <- CitesScholar("Albert Einstein")

  # Check if the final output is a list
  expect_that( ResultsList, is_a("list") )

  # Check its length
  expect_that( length(ResultsList), equals(2) )

  # Check the types of both outputs
  expect_that( ResultsList[[1]], is_a("character") )
  expect_that( ResultsList[[2]], is_a("HTMLInternalDocument") )
})
```

2. Here we test how the function handles the situation where the given name is valid but it is not registered in Google Scholar.

```r
  # Testing the CitesScholar function (input = HTML, output = list)
test_that("Output when a non-existent author name is provided", {
  # A non-registered user name is provided
  ResultsList <- CitesScholar("Cristobal Pais")
})
```

3. Finally, we perform a series of tests for checking how our function handles different non-valid input situations.

```r
# Testing the CitesScholar function for different inputs
test_that("Only first name", {
  # One word is provided
  ResultsList <- CitesScholar("Albert")
})

test_that("More than two names", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("Albert Einstein Canalejo")
})

test_that("Words and numbers", {
  # First and last name with a number
  ResultsList <- CitesScholar("Albert Einstein2")
})

test_that("Words and numbers (mixed)", {
  # Numbers in between characters
  ResultsList <- CitesScholar("Albert E1nstein")
})

test_that("No space between names", {
  # First and last names are not separated
  ResultsList <- CitesScholar("AlbertEinstein")
})

test_that("Uppercase name", {
  # Name is written in uppercase
  ResultsList <- CitesScholar("ALBERT EINSTEIN")
})

test_that("Lowercase name", {
  # Name is written in lowercase
  ResultsList <- CitesScholar("albert einstein")
})

test_that("Erroneous input is provided (numbers)", {
  # Numeric input
  ResultsList <- CitesScholar(1123)
})

test_that("Erroneous input is provided (digit strings)", {
  # String with digits
  FinalDF2 <- CitesScholar("1222")
})
```

In order to run these tests, we can use the following simple code:

```r
# Loading the library
suppressMessages(library(testthat))

# Set working directory
setwd("C:/Users/chile/ps2/RTests/")
```

```r
# Invoking the test
test_file("test_CitesScholar.R")
```

```
## WW....W123456WWWW789a
## Warnings -----------------------------------------------------------------
## 1. Types and lengths are consistent when using a real author's name (@test_CitesScholar.R#7) - incomp
##
## 2. Types and lengths are consistent when using a real author's name (@test_CitesScholar.R#7) - incomp
##
## 3. Output when a non-existent author name is provided (@test_CitesScholar.R#23) - incomplete final li
##
## 4. Uppercase name (@test_CitesScholar.R#54) - incomplete final line found on 'https://scholar.google
##
## 5. Uppercase name (@test_CitesScholar.R#54) - incomplete final line found on 'https://scholar.google
##
## 6. Lowercase name (@test_CitesScholar.R#59) - incomplete final line found on 'https://scholar.google
##
## 7. Lowercase name (@test_CitesScholar.R#59) - incomplete final line found on 'https://scholar.google
##
## Failed -------------------------------------------------------------------
## 1. Error: Output when a non-existent author name is provided (@test_CitesScholar.R#23)
## The author does not exist in the Google Scholar Database, please try again with a different name
## 1: CitesScholar("Cristobal Pais") at test_CitesScholar.R:23
## 2: stop(paste("The author does not exist in the Google Scholar Database,", "please try again with a d
##        sep = " "))
##
## 2. Error: Only first name (@test_CitesScholar.R#29) -----------------------
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("Albert") at test_CitesScholar.R:29
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 3. Error: More than two names (@test_CitesScholar.R#34) -------------------
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("Albert Einstein Canalejo") at test_CitesScholar.R:34
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 4. Error: Words and numbers (@test_CitesScholar.R#39) ---------------------
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("Albert Einstein2") at test_CitesScholar.R:39
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 5. Error: Words and numbers (mixed) (@test_CitesScholar.R#44) -------------
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("Albert E1nstein") at test_CitesScholar.R:44
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 6. Error: No space between names (@test_CitesScholar.R#49) ----------------
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("AlbertEinstein") at test_CitesScholar.R:49
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 7. Error: Erroneous input is provided (numbers) (@test_CitesScholar.R#64) -
## The input must be a name (e.g Geoffrey Hinton)
## 1: CitesScholar(1123) at test_CitesScholar.R:64
## 2: stop("The input must be a name (e.g Geoffrey Hinton)")
##
## 8. Error: Erroneous input is provided (digit strings) (@test_CitesScholar.R#69)
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("1222") at test_CitesScholar.R:69
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 9. Error: Erroneous input is provided (digit strings) (@test_CitesScholar.R#74)
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("1222") at test_CitesScholar.R:74
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## 10. Error: Erroneous input is provided (anything) (@test_CitesScholar.R#79)
## Only two words (no digits) are allowed as the author's name, please try again (e.g Geoffrey Hinton)
## 1: CitesScholar("<html><body>") at test_CitesScholar.R:79
## 2: stop(paste("Only two words (no digits) are allowed as the author's name,", "please try again (e.g
##        sep = " "))
##
## DONE ========================================================================
```

Based on the output we can see that we get a series of warnings regarding the end of the html files we are using for testing. This is due to the fact that these files do not end with a \n or a \r\n characters and thus, the system outputs the corresponding warning. Then, we can easily check that all our tests for checking the input valid formats (when no valid ones were provided) ended as expected: stopping the function and displaying an error message to the user regarding the structure of a valid input argument. On the other hand, tests for checking the output types and their format when a valid input name is given are passed without any errors, as expected.

CreateDataFrame_subset Tests

For this function we also perform three main types of tests: (1) Test the function output using a known researcher name as input, (2) Test the function using a non-existent/non-valid Google Scholar author and (3) Check different input possibilities and how our function is handling them.

Important is to note that in this case since the function uses a HTML object as main input, we are using some previously downloaded *html* files using our *DownloadfromScholar()* function presented in section a). This allows us to perform valid and useful tests.

1. We check if the output obtained is a Dataframe, if the length of its columns is valid and if the number of citations does not contain NA values after transforming it to a numeric object. Again, we are using Albert Einstein as our testing researcher with the difference that in this case, we are loading its HTML file, previously downloaded using *DownloadfromScholar()* (see how we are running the tests at the end of this section).

```r
# Loading the function file to be tested
source("CreateDataFrame_subset.R")

# Testing the CreateDataFrame_subset function (input = HTML, output = Dataframe)
test_that("Types and lengths are consistent when using a known html file", {
  # Save the output in the FinalDF variable, reading a html file from Google Scholar
  # As an example, we are using the citation website of Albert Einstein
  FinalDF <- CreateDataFrame_subset(htmlParse(readLines("~/ps2/download/
                                                        qc6CJjYAAAAJ.html")))

  # Check if the final output is a DataFrame
  expect_that( FinalDF, is_a("data.frame") )

  # Check if all columns have the same length
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Authors)) )
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Year)) )
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Citations)) )
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Journals)) )

  # Check if the Citations column has only numbers
  expect_that( as.numeric(FinalDF$Citations), is_a("numeric") )

})
```

2. Here we test how the function handles the situation where the given HTML is not valid (does not contain the information expected by the function).

```r
# Testing the CreateDataFrame_subset function for different inputs
test_that("Output when a non-existent user html file is provided", {
  # Save the output in the FinalDF variable, after reading a .html
  # file from the Google Scholar website, but from a non-existing user
  FinalDF <- CreateDataFrame_subset(htmlParse(readLines("~/ps2/download/
                                                     nonexistent.html")))
})

# Another html file (not from Google Scholar)
test_that("A html file/code from any page", {
  # HTML downloaded from google.com
  FinalDF <- CreateDataFrame_subset(htmlParse(readLines("~/ps2/download
                                                     /index.html")))
})
```

3. Finally, we perform a series of tests for checking how our function handles different non-valid input situations.

```r
# Numbers as inputs
test_that("Erroneous input is provided (numbers)", {
  # Testing with a numeric input
  FinalDF2 <- CreateDataFrame_subset(1222)
})

# String with numbers
```

```r
test_that("Erroneous input is provided (digit strings)", {
  # String composed by digits
  FinalDF2 <- CreateDataFrame_subset("1222")
})

# Any random string
test_that("Erroneous input is provided (character strings)", {
  # Characters string
  FinalDF2 <- CreateDataFrame_subset("hello")
})
```

In order to run these tests, we can use the following simple code.

```r
# Downloading the files (using wget for simplicity): valid, non-existent and non-valid
system('bash -c "wget -q -O qc6CJjYAAAAJ.html "https://scholar.google.com/"\
"citations?user=qc6CJjYAAAAJ&hl=es&oi=ao""')
system('bash -c "wget -q -O nonexistent.html "https://scholar.google.com/scholar?"\
"q=Cristobal+Pais&btnG=&hl=es&as_sdt=0%2C5""')
system('bash -c "wget -q www.google.com"')
```

```r
# Loading the library
suppressMessages(library(testthat))

# Set working directory
setwd("C:/Users/chile/ps2/RTests/")

# Invoking the test
test_file("test_CreateDataFrame_subset.R")
```

```
## WW......1W23456
## Warnings -----------------------------------------------------------------
## 1. Types and lengths are consistent when using a known html file (@test_CreateDataFrame_subset.R#8)
##
## 2. Types and lengths are consistent when using a known html file (@test_CreateDataFrame_subset.R#8)
##
## 3. A html file/code from any page (@test_CreateDataFrame_subset.R#33) - incomplete final line found
##
## Failed -------------------------------------------------------------------
## 1. Error: Output when a non-existent user html file is provided (@test_CreateDataFrame_subset.R#27)
## argumento tiene longitud cero
## 1: CreateDataFrame_subset(htmlParse(readLines("nonexistent.html"))) at test_CreateDataFrame_subset.R
## 2: typeof(html)
## 3: htmlParse(readLines("nonexistent.html"))
##
## 2. Error: A html file/code from any page (@test_CreateDataFrame_subset.R#33)
## No Titles: There is no author's citation page inside Google Scholar associated with the html file be
## 1: CreateDataFrame_subset(htmlParse(readLines("index.html"))) at test_CreateDataFrame_subset.R:33
## 2: stop(paste("No Titles: There is no author's citation page inside Google Scholar",
##        "associated with the html file being processed, please", "try again with a different file",
##        sep = " "))
##
## 3. Error: Erroneous input is provided (numbers) (@test_CreateDataFrame_subset.R#39)
## Not a valid html file from author's citation website
## 1: CreateDataFrame_subset(1222) at test_CreateDataFrame_subset.R:39
## 2: stop("Not a valid html file from author's citation website")
##
## 4. Error: Erroneous input is provided (digit strings) (@test_CreateDataFrame_subset.R#45)
## Not a valid html file from author's citation website
## 1: CreateDataFrame_subset("1222") at test_CreateDataFrame_subset.R:45
## 2: stop("Not a valid html file from author's citation website")
##
## 5. Error: Erroneous input is provided (character strings) (@test_CreateDataFrame_subset.R#51)
## Not a valid html file from author's citation website
## 1: CreateDataFrame_subset("hello") at test_CreateDataFrame_subset.R:51
## 2: stop("Not a valid html file from author's citation website")
##
## 6. Error: Erroneous input is provided (html sintax) (@test_CreateDataFrame_subset.R#57)
## Not a valid html file from author's citation website
## 1: CreateDataFrame_subset("<html><body>") at test_CreateDataFrame_subset.R:57
## 2: stop("Not a valid html file from author's citation website")
##
## DONE ======================================================================
```

Based on the output we can easily check that we again get some warnings regarding the end of the HTML files and the NAs values introduced by our function when we are cleaning the Citations array. None of these warnings are relevant for our implementation. Then, we can clearly see that all the tests developed when a valid input is given to the function (Albert Einstein html file in this example) are passed without any problem and then, we get a series of errors when non-valid input arguments are passed to our function: non-existing author query, numbers, strings containing numbers, a random string, and non-valid html files.

d) Extra Credit: All citations as DataFrame

Based on the code developed for section b), we just need to add some simple modifications in order to download all the author's articles. In this case, we will perform a series of queries based on our knowledge of how Google Scholar implements the "show more" logic when hitting the button at the bottom of the author's personal website, that will allow us to extract different HTML files structures, each one associated with (at most) one hundred publications. In order to obtain this information, we simply used the developer tools available in Google Chrome and we check the structure of the query that is triggered when we press the button.

The structure of the query consists of a starting and size values: the start value indicates the number of the article from which the query will start displaying minus 1 (e.g start = 0 implies displaying from the first article, start = 100 then 101, and so on) and the size value indicates the maximum number of articles that can be displayed at the same time (same window) with an internal limit of 100 entries. Hence, we will perform a series of queries of size 100, while changing the start value at every iteration until we obtain an empty query, indicating that no more articles are available for the author.

The function is as follows:

1. As in the original function *CreateDataFrame_subset()* we declare the function, load the libraries and perform the first check of the input, validating its type.

```r
# Declaring the dunction
CreateDataFrame <- function(html){
  # Libraries
  library(XML)
  library(RCurl)
  library(stringr)

  # Turning off the warnings for visualization purposes
  options(warn = -1)

  # Check if the file is a parsed HTML
  if (typeof(html) != "externalptr"){
    stop(paste("Not a valid html file from author's citation website",
               "please check the format of the input file", sep = " "))
  }

   if (class(html)[[1]] != "HTMLInternalDocument") {
    stop(paste("Not a valid html file from author's citation website",
               "please check the format of the input file", sep = " "))
  }
```

2. Then, the first difference consists of the fact that we are extracting (again) the author's personal citation website from the HTML object because we will need it for generating the iterative queries. Note that we could have simply used the user ID and then, based on the known structure of the link, generated it. However, since the function in part b) can only use the HTML object as input (as stated in the question), we are also satisfying that requirement in this case.

   A second check is performed: if there are no links, it means that it is not a valid HTML object for Google Scholar and the function stops.

```r
  # Get the links inside the html
  links <- getHTMLLinks(html, baseURL = urlInit, relative = FALSE)
```

```r
# Check if there are valid links in the file
if (length(links) <= 0){
  stop(paste("Not a valid html file from author's citation website",
             "the file does not contain any valid link", sep = " "))
}

# Generate the link containing the query
url4queries <- paste("http://scholar.google.com", links[8], sep = '')
```

3. We generate five global vectors for our function associated with each of the columns of the desired DataFrame. On the other hand, we initialize the "from and size" values for the first query (0+1 to 100 papers).

```r
# Global vectors for recording all the queries
GTitles <- c()
GJournals <- c()
GAuthors <- c()
GYears <- c()
GCitations <- c()

# Initial values: from (start) and size
from <- 0
size <- 100
```

4. A while loop starts with TRUE as the main condition, allowing us to use a *break* statement inside it. The url containing the query is generated using its known structure and we extract the content of the HTML file on that website. This HTML will contain up to 100 references, depending on the number of valid entries associated with the author's Google Scholar account.

```r
# Loop for getting all the articles based on the query structure
while (TRUE){
  query <- paste("&cstart=",from,"&pagesize=",size,sep = '')
  urlq <- paste(url4queries,query,sep = '')
  html <- readLines(urlq)
  html <- htmlParse(html)
```

5. Same queries as in the original case are performed in order to extract all the relevant information from the HTML object. Important is to note that the *break* condition consists of returning an empty query (no new Titles), indicating that the author does not have more articles published and linked to the Google Scholar engine.

```r
  # Articles titles
  Titles <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                        tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_t']
                        //a[@class = 'gsc_a_at']", xmlValue,
                        trim = TRUE, encoding = "UTF-8")

  # Break condition: no titles/papers inside the HTML object
  if (length(Titles) == 0 ) {
    break
  }

  # Journals and authors array
  JournalAuthors <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                              tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_t']
                              //div[@class = 'gs_gray']", xmlValue,
                              trim = TRUE, encoding = "UTF-8")

  # Getting the Journals (even numbers of previos vector)
  Journals <- JournalAuthors[seq(2, length(JournalAuthors), 2)]

  # We delete the year of publication after the Journal
  Journals <- substring(Journals, 0, nchar(Journals)-6)

  # Getting the authors (odd numbers)
  Authors <- JournalAuthors[seq(1, length(JournalAuthors), 2)]

  # Years of publication
  Year <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//tr[@class ='gsc_a_tr']
                    //td[@class = 'gsc_a_y']//span[@class = 'gsc_a_h']",
                    xmlValue, trim = TRUE, encoding = "UTF-8")

  # Number of citations (No repetitions)
  Citations <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                         tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_c']//a",
                         xmlValue, trim = TRUE, encoding = "UTF-8")

  # Delete potential non numerical elements by transforming and filtering the array
  Citations <- as.numeric(Citations)
  remove <- c(NA)

  # Re-Transform it to a character array for the DF
  Citations <- as.character(Citations[!Citations %in% remove])
```

6. We update the content of the global vectors. Note that we are appending the vectors obtained from the new queries in order to preserve all values per iteration (not replacing them). We end the loop by updating the from value by 100 in order to perform a new query including the next (up to) 100 entries. Note that we can add a command like Sys.sleep(2) in between of the iterations in order to not perform a series of queries without some delay between them (good webscraping practices, as suggested in class).

```
  # Record in global vectors
  GTitles <- c(GTitles, Titles)
  GAuthors <- c(GAuthors, Authors)
  GJournals <- c(GJournals, Journals)
  GYears <- c(GYears, Year)
  GCitations <- c(GCitations, Citations)

  # Update the counter
  from <- from + 100
}
```

7. As in the original function, we perform a length check regarding the number of citations vector. We fill it with as many zeros as needed in order to match the length of the other vectors. This can be done without any inconsistency since the articles are ordered by number of citations starting with the highest one.

```
  # Dimension checks: fill with 0 empty citations
  if (length(GCitations) != length(GTitles)){
    GCitations <- c(GCitations, integer(length(GTitles) - length(GCitations)))
  }
```

8. The DataFrame information and labels are defined. The complete DataFrame (with all citations) is returned to the user.

```
  # Create a DataFrame
  x_name <- "Title"
  y_name <- "Authors"
  z_name <- "Journals"
  w_name <- "Year"
  v_name <- "Citations"

  require(reshape2)
  ScholarDF <- melt(data.frame(GTitles, GAuthors, GJournals, GYears, GCitations))
  colnames(ScholarDF) <- c(x_name, y_name, z_name, w_name, v_name)

  # Return the DataFrame
  return(ScholarDF)
}
```

Thus, we extended our original function such that the new one uses the same HTML object as input but returns all the results for a researcher instead of only the first 20. We can easily run the function using the following code:

```
# Call the function and create a DataFrame from the author's website
ResultsList <- CitesScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Create the full data frame
FullScholarDF <- CreateDataFrame(Authorhtml)

# Display the length of the DataFrame (checking the length of any column)
length(FullScholarDF$Title)
```

```
## [1] 553
```

```
# Printing a summary using the apply function (``pretty'' output)
apply(FullScholarDF, 2, function(x)
  if (length(x) > 15 && is.na(str_match(x, "^[[:digit:]]"))) {
    paste(substr(x, start = 0, stop = 15), "...", sep = "")
  }
  else {substr(x, start = 0, stop = length(x))
  }
)
```

```
##        Title                            Authors
##    [1,] "Learning repres..."        "DE Rumelhart, G..."
##    [2,] "Learning intern..."        "DE Rumelhart, G..."
##    [3,] "Learning intern..."        "DE Rumelhart, G..."
##    [4,] "Parallel distri..."        "DE Rumelhart, J..."
##    [5,] "Imagenet classi..."        "A Krizhevsky, I..."
##    [6,] "A fast learning..."        "GE Hinton, S Os..."
##    [7,] "Parallel distri..."        "JL McClelland, ..."
##    [8,] "Reducing the di..."        "GE Hinton, RR S..."
##    [9,] "Deep learning..."          "Y LeCun, Y Beng..."
##   [10,] "Adaptive mixtur..."        "RA Jacobs, MI J..."
##   [11,] "Dropout: a simp..."        "N Srivastava, G..."
##   [12,] "A learning algo..."        "DH Ackley, GE H..."
##   [13,] "Visualizing dat..."        "L van der Maate..."
##   [14,] "Deep neural net..."        "G Hinton, L Den..."
##   [15,] "Training produc..."        "GE Hinton..."
##   [16,] "A view of the E..."        "RM Neal, GE Hin..."
##   [17,] "Phoneme recogni..."        "A Waibel, T Han..."
##   [18,] "Improving neura..."        "GE Hinton, N Sr..."
##   [19,] "Rectified linea..."        "V Nair, GE Hint..."
##   [20,] "Connectionist l..."        "GE Hinton..."
##   [21,] "Learning and re..."        "GE Hinton, TJ S..."
##   [22,] "Distributed rep..."        "GE Hinton, JL M..."
##   [23,] "Learning multip..."        "A Krizhevsky, G..."
##   [24,] "Speech recognit..."        "A Graves, A Moh..."
##   [25,] "Schemata and se..."        "D Rumelhart, P ..."
##   [26,] "How learning ca..."        "GE Hinton, SJ N..."
##   [27,] "Parallel models..."        "GE Hinton, JA A..."
##   [28,] "Neighbourhood c..."        "J Goldberger, G..."
##   [29,] "Deep Boltzmann ..."        "R Salakhutdinov..."
##   [30,] "A practical gui..."        "G Hinton..."
##   [31,] "Acoustic modeli..."        "A Mohamed, G Da..."
##   [32,] "Restricted Bolt..."        "R Salakhutdinov..."
##   [33,] "The appeal of p..."        "JL McClelland, ..."
##   [34,] "The microarchit..."        "G Hinton, D Sag..."
##   [35,] "The\" wake-sleep..."       "GE Hinton, P Da..."
##   [36,] "Lesioning an at..."        "GE Hinton, T Sh..."
##   [37,] "The helmholtz m..."        "P Dayan, GE Hin..."
##   [38,] "Learning distri..."        "GE Hinton..."
##   [39,] "The EM algorith..."        "Z Ghahramani, G..."
##   [40,] "Lecture 6.5-rms..."        "T Tieleman, G H..."
##   [41,] "Stochastic neig..."        "GE Hinton, ST R..."
##   [42,] "On the importan..."        "I Sutskever, J ..."
##   [43,] "Semantic hashin..."        "R Salakhutdinov..."
##   [44,] "Learning repres..."        "D Williams, G H..."
##   [45,] "Learning multip..."        "GE Hinton..."
##   [46,] "Boltzmann machi..."        "GE Hinton, TJ S..."
##   [47,] "A time-delay ne..."        "KJ Lang, AH Wai..."
##   [48,] "Optimal percept..."        "GE Hinton, TJ S..."
##   [49,] "A scalable hier..."        "A Mnih, GE Hint..."
##   [50,] "A practical gui..."        "GE Hinton..."
##   [51,] "How neural netw..."        "GE Hinton..."
##   [52,] "Classical and B..."        "KJ Friston, W P..."
##   [53,] "Modeling human ..."        "GW Taylor, GE H..."
##   [54,] "Parameter estim..."        "Z Ghahramani, G..."
##  [55,] "Generating text..."        "I Sutskever, J ..."
##  [56,] "Distilling the ..."        "G Hinton, O Vin..."
##  [57,] "Simplifying neu..."        "SJ Nowlan, GE H..."
##  [58,] "Implementing se..."        "GE Hinton..."
##  [59,] "Improving deep ..."        "GE Dahl, TN Sai..."
##  [60,] "On contrastive ..."        "MA Carreira-Per..."
##  [61,] "Keeping the neu..."        "GE Hinton, D Va..."
##  [62,] "Experiments on ..."        "DC Plaut..."
##  [63,] "SMEM algorithm ..."        "N Ueda, R Nakan..."
##  [64,] "Parallel visual..."        "DH Ballard, GE ..."
##  [65,] "Glove-talk: A n..."        "SS Fels, GE Hin..."
##  [66,] "A distributed c..."        "DS Touretzky, G..."
##  [67,] "Autoencoders, m..."        "GE Hinton, RS Z..."
##  [68,] "Modeling the ma..."        "GE Hinton, P Da..."
##  [69,] "Exponential fam..."        "M Welling, M Ro..."
##  [70,] "Three new graph..."        "A Mnih, G Hinto..."
##  [71,] "Products of exp..."        "GE Hinton..."
##  [72,] "Feudal reinforc..."        "P Dayan, GE Hin..."
##  [73,] "Self-organizing..."        "S Becker, GE Hi..."
##  [74,] "Zero-shot learn..."        "M Palatucci, D ..."
##  [75,] "Efficient learn..."        "R Salakhutdinov..."
##  [76,] "Learning a nonl..."        "R Salakhutdinov..."
##  [77,] "Deep belief net..."        "A Mohamed, G Da..."
##  [78,] "Mapping part-wh..."        "GE Hinton..."
##  [79,] "Neuroanimator: ..."        "R Grzeszczuk, D..."
##  [80,] "MASSIVELY PARAL..."        "SE Fahlman, GE ..."
##  [81,] "Preface to the ..."        "GE Hinton..."
##  [82,] "An alternative ..."        "L Xu, MI Jordan..."
##  [83,] "Variational lea..."        "Z Ghahramani, G..."
##  [84,] "Generative mode..."        "GE Hinton, Z Gh..."
##  [85,] "A time-delay ne..."        "KJ Lang..."
##  [86,] "Replicated soft..."        "GE Hinton, RR S..."
##  [87,] "Unsupervised le..."        "GE Hinton, TJ S..."
##  [88,] "New types of de..."        "L Deng, G Hinto..."
##  [89,] "3D object recog..."        "V Nair, GE Hint..."
##  [90,] "Symbols among t..."        "DS Touretzky, G..."
##  [91,] "Models of infor..."        "JA Anderson, GE..."
##  [92,] "Grammar as a fo..."        "O Vinyals, L Ka..."
##  [93,] "A parallel comp..."        "GE Hinton..."
##  [94,] "Factored condit..."        "GW Taylor, GE H..."
##  [95,] "On rectified li..."        "MD Zeiler, M Ra..."
##  [96,] "To recognize sh..."        "GE Hinton..."
##  [97,] "Global coordina..."        "ST Roweis, LK S..."
##  [98,] "Parallel comput..."        "G Hinton..."
##  [99,] "Using fast weig..."        "T Tieleman, G H..."
## [100,] "Phone recogniti..."        "G Dahl, A Moham..."
## [101,] "The recurrent t..."        "I Sutskever, GE..."
## [102,] "A theoretical f..."        "Y Le Cun, D Tou..."
## [103,] "Phoneme recogni..."        "A Waibel, T Han..."
## [104,] "Connectionist a..."        "SE Fahlman, GE ..."
## [105,] "Modeling pixel ..."        "GE Hinton..."
## [106,] "Some demonstrat..."        "G Hinton..."
## [107,] "Deep belief net..."        "A Mohamed, TN S..."
## [108,] "Deterministic B..."        "GE Hinton..."
## [109,] "Shape represent..."        "GE Hinton..."
## [110,] "Learning to rep..."        "R Memisevic, GE..."
## [111,] "Semantic hashin..."        "R Salakhutdinov..."
## [112,] "Learning transl..."        "GE Hinton..."
## [113,] "Glove-TalkII-a ..."        "SS Fels, GE Hin..."
## [114,] "Factored 3-way ..."        "A Krizhevsky, G..."
## [115,] "Binary coding o..."        "L Deng, ML Selt..."
## [116,] "Using generativ..."        "M Revow, CKI Wi..."
## [117,] "Understanding h..."        "A Mohamed, G Hi..."
## [118,] "Learning to det..."        "V Mnih, GE Hint..."
## [119,] "Neurocomputing:..."        "DE Rumelhart, G..."
## [120,] "Learning multil..."        "I Sutskever, G ..."
## [121,] "Using very deep..."        "A Krizhevsky, G..."
## [122,] "Learning sparse..."        "M Welling, S Os..."
## [123,] "Variational lea..."        "Z Ghahramani, G..."
## [124,] "Varieties of He..."        "P Dayan, GE Hin..."
## [125,] "On deep generat..."        "J Susskind, V M..."
## [126,] "A simple way to..."        "QV Le, N Jaitly..."
## [127,] "Unsupervised le..."        "R Memisevic, G ..."
## [128,] "Using expectati..."        "P Dayan, GE Hin..."
## [129,] "Learning repres..."        "GE Hinton, JL M..."
## [130,] "Transforming au..."        "GE Hinton, A Kr..."
## [131,] "Adaptive elasti..."        "GE Hinton, CKI ..."
## [132,] "Separating figu..."        "PK Kienker, TJ ..."
## [133,] "Energy-based mo..."        "YW Teh, M Welli..."
## [134,] "Learning sets o..."        "DC Plaut, GE Hi..."
## [135,] "Learning symmet..."        "TJ Sejnowski, P..."
## [136,] "Convolutional d..."        "A Krizhevsky, G..."
## [137,] "Recognizing han..."        "GE Hinton, M Re..."
## [138,] "Using fast weig..."        "GE Hinton, DC P..."
## [139,] "Using deep beli..."        "GE Hinton, RR S..."
## [140,] "Deep belief net..."        "GE Hinton..."
## [141,] "Analyzing coope..."        "GE Hinton, TJ S..."
## [142,] "Application of ..."        "R Sarikaya, GE ..."
## [143,] "Evaluation of a..."        "SJ Nowlan, GE H..."
## [144,] "Representing pa..."        "GE Hinton..."
## [145,] "Frames of refer..."        "GE Hinton, LM P..."
## [146,] "Scene-based and..."        "GE Hinton, LM P..."
## [147,] "A new learning ..."        "M Welling, GE H..."
## [148,] "Relaxation and ..."        "GE Hinton..."
## [149,] "Layer normaliza..."        "JL Ba, JR Kiros..."
## [150,] "Learning to com..."        "H Larochelle, G..."
## [151,] "Modeling image ..."        "S Osindero, GE ..."
## [152,] "Learning to rep..."        "GE Hinton..."
## [153,] "Dynamical binar..."        "GW Taylor, L Si..."
## [154,] "Topographic pro..."        "S Osindero, M W..."
## [155,] "Rate-coded rest..."        "YW Teh, GE Hint..."
## [156,] "Reinforcement l..."        "B Sallans, GE H..."
## [157,] "Shape recogniti..."        "GE Hinton, KJ L..."
## [158,] "Learning a bett..."        "N Jaitly, G Hin..."
## [159,] "Learning in par..."        "G Hinton..."
## [160,] "Modeling docume..."        "N Srivastava, R..."
## [161,] "Learning to lab..."        "V Mnih, GE Hint..."
## [162,] "Simulating brai..."        "GE Hinton, DC P..."
## [163,] "Switching state..."        "Z Ghahramani, G..."
## [164,] "Robust boltzman..."        "Y Tang, R Salak..."
## [165,] "Separating figu..."        "TJ Sejnowski, G..."
## [166,] "Split and merge..."        "N Ueda, R Nakan..."
## [167,] "Glove-TalkII: a..."        "S Fels, G Hinto..."
## [168,] "Deep, narrow si..."        "I Sutskever, GE..."
## [169,] "Automatic recog..."        "DH Pruslin, G R..."
## [170,] "Conditional Res..."        "V Mnih, H Laroc..."
## [171,] "A mobile robot ..."        "S Oore, GE Hint..."
## [172,] "Developing popu..."        "RS Zemel, GE Hi..."
## [173,] "A comparison of..."        "M Ennis, G Hint..."
## [174,] "Recognizing han..."        "G Mayraz, GE Hi..."
## [175,] "A soft decision..."        "SJ Nowlan, GE H..."
## [176,] "Two distributed..."        "GW Taylor, GE H..."
## [177,] "Implicit mixtur..."        "V Nair, GE Hint..."
## [178,] "Phone recogniti..."        "A Mohamed, G Hi..."
## [179,] "Generating faci..."        "JM Susskind, GE..."
## [180,] "Glove-TalkII: M..."        "S Fels, GE Hint..."
## [181,] "GTM through tim..."        "CM Bishop, GE H..."
## [182,] "Discovering bin..."        "G Hinton, R Sal..."
## [183,] "Generating more..."        "M Ranzato, V Mn..."
## [184,] "Inferring motor..."        "V Nair, GE Hint..."
## [185,] "Distinguishing ..."        "CM Bishop, M Sv..."
## [186,] "A better way to..."        "GE Hinton, RR S..."
## [187,] "Deep belief net..."        "R Sarikaya, GE ..."
## [188,] "Generative vers..."        "T Schmah, GE Hi..."
## [189,] "h Williams, R.(..."        "D Rumelhart, G ..."
## [190,] "Visualizing sim..."        "J Cook, I Sutsk..."
## [191,] "Vocal tract len..."        "N Jaitly, GE Hi..."
## [192,] "The appeal of p..."        "JL McClelland, ..."
## [193,] "Building adapti..."        "S Fels, GE Hint..."
## [194,] "Neural networks..."        "G Hinton, N Sri..."
## [195,] "A desktop input..."        "S Oore, D Terzo..."
## [196,] "Visualizing non..."        "L Van der Maate..."
## [197,] "Reinforcement l..."        "B Sallans, GE H..."
## [198,] "Learning mixtur..."        "S Becker, GE Hi..."
## [199,] "A general frame..."        "DE Rumelhart, G..."
## [200,] "Modeling the jo..."        "J Susskind, G H..."
## [201,] "Does the wake-s..."        "BJ Frey, GE Hin..."
## [202,] "Mean field netw..."        "CKI Williams, G..."
## [203,] "Boltzmann machi..."        "G Hinton..."
## [204,] "Unsupervised di..."        "G Hinton, S Osi..."
## [205,] "Learning distri..."        "A Paccanaro, GE..."
## [206,] "Variational lea..."        "BJ Frey, GE Hin..."
## [207,] "A hierarchical ..."        "GE Hinton, B Sa..."
## [208,] "Gated softmax c..."        "R Memisevic, C ..."
## [209,] "The DELVE manua..."        "CE Rasmussen, R..."
## [210,] "G-maximization:..."        "BA Pearlmutter,..."
## [211,] "McClelland.(198..."        "DE Rumelhart..."
## [212,] "Attend, infer, ..."        "SMA Eslami, N H..."
## [213,] "Adaptive soft w..."        "SJ Nowlan, GE H..."
## [214,] "Discovering vie..."        "RS Zemel, GE Hi..."
## [215,] "The bootstrap W..."        "GE Hinton, SJ N..."
## [216,] "A modified gati..."        "L Xu, MI Jordan..."
## [217,] "Evaluating the ..."        "N Hammond, G Hi..."
## [218,] "Spatial Coheren..."        "GE Hinton..."
## [219,] "Solving random-..."        "R Szeliski, G H..."
## [220,] "Where do featur..."        "G Hinton..."
## [221,] "Products of Hid..."        "AD Brown, GE Hi..."
## [222,] "Spiking boltzma..."        "GE Hinton, A Br..."
## [223,] "Self supervised..."        "M Welling, RS Z..."
## [224,] "Some computatio..."        "G Hinton..."
## [225,] "Comparing class..."        "T Schmah, G You..."
## [226,] "What kind of gr..."        "GE Hinton..."
## [227,] "Learning to par..."        "GE Hinton, Z Gh..."
## [228,] "Advances in neu..."        "C Linster, D Ma..."
## [229,] "Deep mixtures o..."        "Y Tang, R Salak..."
## [230,] "Multiple relati..."        "R Memisevic, GE..."
## [231,] "Delve data for ..."        "CE Rasmussen, R..."
## [232,] "Dimensionality ..."        "KJ Lang, GE Hin..."
## [233,] "Modeling natura..."        "V Mnih, JM Suss..."
## [234,] "Local physical ..."        "S Oore, D Terzo..."
## [235,] "Discovering mul..."        "GE Hinton, YW T..."
## [236,] "Deep lambertian..."        "Y Tang, R Salak..."
## [237,] "A 0.18/spl mu/m..."        "D Sager, G Hint..."
## [238,] "Hierarchical no..."        "Z Ghahramani, G..."
## [239,] "A general frame..."        "DE Rumelhart, G..."
## [240,] "Outrageously la..."        "N Shazeer, A Mi..."
## [241,] "Tensor analyzer..."        "Y Tang, R Salak..."
## [242,] "Using free ener..."        "B Sallans, GE H..."
## [243,] "Learning genera..."        "N Heess, C Will..."
## [244,] "Combining Defor..."        "CKI Williams, G..."
## [245,] "TRAFFIC: Recogn..."        "RS Zemel, MC Mo..."
## [246,] "Learning intern..."        "DE Rumelhart, G..."
## [247,] "Learning Causal..."        "GE Hinton, S Os..."
## [248,] "Why the islands..."        "E Hutchins, GE ..."
## [249,] "Lecture 6a over..."        "G Hinton, N Sri..."
## [250,] "Psychology will..."        "S Scarr..."
## [251,] "Schemata and se..."        "DE Rumelhart, P..."
## [252,] "Temporal-kernel..."        "I Sutskever, G ..."
## [253,] "A new view of I..."        "GE Hinton, M We..."
## [254,] "An unsupervised..."        "GE Hinton, S Be..."
## [255,] "Introduction to..."        "D Yu, G Hinton,..."
## [256,] "Keeping neural ..."        "GE Hinton, D va..."
## [257,] "Learning intern..."        "DE Rumelhart, G..."
## [258,] "Varieties of He..."        "GE Hinton, P Da..."
## [259,] "G. and Williams..."        "DH Rumelhart, G..."
## [260,] "Imagery without..."        "G Hinton..."
## [261,] "On the importan..."        "I Sutskever, J ..."
## [262,] "The ups and dow..."        "G Hinton..."
## [263,] "Coaching variab..."        "R Tibshirani, G..."
## [264,] "A simple algori..."        "BJ Frey, P Daya..."
## [265,] "Connectionist m..."        "DS Touretzky, J..."
## [266,] "Autoregressive ..."        "N Jaitly, V Van..."
## [267,] "Analysis-by-syn..."        "V Nair, J Sussk..."
## [268,] "Neighbourhood c..."        "S Roweis, G Hin..."
## [269,] "Deterministic B..."        "CC Galland, GE ..."
## [270,] "GEMINI: gradien..."        "Y Le Cun, CC Ga..."
## [271,] "Deep learning f..."        "L Deng, D Yu, G..."
## [272,] "Instantiating d..."        "CKI Williams, M..."
## [273,] "Using relaxatio..."        "GE Hinton..."
## [274,] "Using an autoen..."        "N Jaitly, GE Hi..."
## [275,] "A mode-hopping ..."        "C Sminchisescu,..."
## [276,] "Learning sequen..."        "DE Rumelhart, G..."
## [277,] "e Willians, R.(..."        "DE Rumelhart, G..."
## [278,] "Improving a sta..."       "A Mnih, Z Yuech..."
## [279,] "Generalized dar..."        "C Sminchisescu,..."
## [280,] "Learning hierar..."        "APGE Hinton..."
## [281,] "Using pairs of ..."        "GE Hinton, M Re..."
## [282,] "Learning intern..."        "DE Rumelhart, G..."
## [283,] "The role of spa..."        "GE Hinton..."
## [284,] "Deep belief net..."        "G Hinton..."
## [285,] "Wormholes impro..."        "G Hinton, M Wel..."
## [286,] "Extracting dist..."        "A Paccanaro, GE..."
## [287,] "Une nouvelle ap..."        "JL McClelland, ..."
## [288,] "Learning intern..."        "D Rumelhart, G ..."
## [289,] "Learning Intern..."        "DE Rumelhart, G..."
## [290,] "Parallel Distri..."        "DE Rumelhart, G..."
## [291,] "Learning intern..."        "DE Rumelhart, G..."
## [292,] "Products of hid..."        "GW Taylor, GE H..."
## [293,] "Using matrices ..."        "I Sutskever, GE..."
## [294,] "Improving dimen..."        "R Memisevic, G ..."
## [295,] "Learning in Gra..."        "R Neal, GE Hint..."
## [296,] "Efficient stoch..."        "BJ Frey, GE Hin..."
## [297,] "Learning intern..."        "GE Hinton, DE R..."
## [298,] "Inferring the m..."        "GE Hinton..."
## [299,] "Modeling pigeon..."        "MD Zeiler, GW T..."
## [300,] "Neural network-..."        "DA Eichmann, K ..."
## [301,] "TRAFFIC: A mode..."        "RS Zemel, MC Mo..."
## [302,] "Regularizing ne..."        "G Pereyra, G Tu..."
## [303,] "A better way to..."        "GE Hinton..."
## [304,] "How to do backp..."        "G Hinton..."
## [305,] "How neural netw..."        "GE Hinton..."
## [306,] "Connectionist l..."        "GE Hinton..."
## [307,] "Data for evalua..."        "CE Rasmussen, R..."
## [308,] "Minimizing desc..."        "GE Hinton, RS Z..."
## [309,] "The Helmholtz m..."        "G Hinton, P Day..."
## [310,] "Relaxing the hy..."        "LY Pratt, AN Ch..."
## [311,] "Using mixtures ..."        "M Revow, CKI Wi..."
## [312,] "Schemata and se..."        "DE Rumlehart, P..."
## [313,] "E., McClelland,..."        "D Rumelhart, GE..."
## [314,] "A new way to le..."        "N Jaitly, GE Hi..."
## [315,] "Computation by ..."        "GE Hinton..."
## [316,] "Technical Repor..."        "Z Ghahramani, G..."
## [317,] "Hand-printed di..."        "CKI Williams, M..."
## [318,] "Discovering hig..."        "CC Galland, GE ..."
## [319,] "Readings in spe..."        "A Waibel, T Han..."
## [320,] "Pattern matchin..."        "DS Touretzky, G..."
## [321,] "Learning Intern..."        "DE Rumelhart, G..."
## [322,] "Learning intern..."        "DE Rumelhart, G..."
## [323,] "PDP: Computatio..."        "DE Rumelhart, G..."
## [324,] "Novice use of i..."        "N Hammond, A Ma..."
## [325,] "Guest editorial..."        "G Hinton, Y LeC..."
## [326,] "Learning nonlin..."        "A Mnih, G Hinto..."
## [327,] "Free energy cod..."        "BJ Frey, GE Hin..."
## [328,] "Using a neural ..."        "CKI Williams, M..."
## [329,] "Phoneme recogni..."        "G Hinton, A Wai..."
## [330,] "Learning Intern..."        "DE Rumelhart, G..."
## [331,] "Learning Intern..."        "GE Hinton, DE R..."
## [332,] "Learning intern..."        "DE Rumelhart, G..."
## [333,] "K., Lang, K., D..."        "A Waibel, H Tos..."
## [334,] "Machine learnin..."        "GE Hinton..."
## [335,] "Probabilistic s..."        "M Welling, RS Z..."
## [336,] "Learning to mak..."        "S Becker, GE Hi..."
## [337,] "Learning in mas..."        "GE Hinton..."
## [338,] "Learning Intern..."        "DE Rummelhart, ..."
## [339,] "Learning intern..."        "JW Ronald, DE R..."
## [340,] "Representation ..."        "A Sloman, D Owe..."
## [341,] "Sejnows ki, TJ ..."        "DH Ackley, GA H..."
## [342,] "Using fast weig..."        "J Ba, GE Hinton..."
## [343,] "Dark knowledge..."         "GE Hinton, O Vi..."
## [344,] "Generative mult..."        "T Adel, B Smith..."
## [345,] "Training many s..."        "GE Hinton, AD B..."
## [346,] "Modeling High-D..."        "GE Hinton..."
## [347,] "Learning Distri..."        "A Paccanaro, GE..."
## [348,] "Delve data sets..."        "CE Rasmussen, R..."
## [349,] "El atractivo de..."        "JL McClelland, ..."
## [350,] "Combining two m..."        "G Hinton, C Wil..."
## [351,] "Generative back..."        "GE Hinton..."
## [352,] "The horizontal<U+0097>..."        "GE Hinton..."
## [353,] "Phoneme recogni..."        "A Waibel, T Han..."
## [354,] "Parallel Data P..."        "D Rumelhart, J ..."
## [355,] "8: McClelland, ..."        "D Rumelhart, GE..."
## [356,] "Learning Intern..."        "E Rumelhart, GE..."
## [357,] "Affinity weight..."        "J Weston, R Wei..."
## [358,] "Neural Networks..."        "G Hinton, N Sri..."
## [359,] "Antibacterial p..."        "L Tahir, N Khan..."
## [360,] "Improving a sta..."        "Z Yuecheng, A M..."
## [361,] "The EM algorith..."        "Z Ghahramani, G..."
## [362,] "Neural networks..."        "SS Fels, G Hint..."
## [363,] "and University ..."        "L Cooper, J Cow..."
## [364,] "Neural network ..."        "GE Hinton..."
## [365,] "A general frame..."        "DE Rumelhart, G..."
## [366,] "Learning intern..."        "DE Rumelhart, G..."
## [367,] "Learning Intern..."        "DE Rumelhart, G..."
## [368,] "Learning repres..."        "DE Ramelhart, G..."
## [369,] "Learning intern..."        "DE Rumelhart, G..."
## [370,] "Learning repres..."        "D Rummelhart, G..."
## [371,] "Parallel Distri..."        "DE Rumelhart, G..."
## [372,] "Learning Intern..."        "DE Rummelhart, ..."
## [373,] "Learning intern..."        "DE Rumelhart, G..."
## [374,] "Parallel Distri..."        "DE Rumelhart, G..."
## [375,] "Williams (1986)..."        "DE Rumelhart, G..."
## [376,] "McClelland.(198..."        "DE Rumelhart, G..."
## [377,] "8: McClel1and, ..."        "DE Rumelhart, G..."
## [378,] "Delve data for ..."        "CE Rasmussen, R..."
## [379,] "Efficient Param..."        "MWRSZ Geoffrey,..."
## [380,] "Rectified linea..."        "GE Hinton..."
## [381,] "The next genera..."        "G Hinton..."
## [382,] "Deep belief net..."        "R Salakhutdinov..."
## [383,] "Relative densit..."        "AD Brown, GE Hi..."
## [384,] "Scaling in a hi..."        "Z Ghahramani, A..."
## [385,] "A View of the E..."        "M Neal Radford..."
## [386,] "Cascaded redund..."        "VR De Sa, GE Hi..."
## [387,] "Learning Error ..."        "DE Rumelhart, G..."
## [388,] "the PDP Researc..."        "DE Runmelhart, ..."
## [389,] "Models of schem..."        "DE Rumelhart, P..."
## [390,] "Reducing the Di..."        "GE Hinton, RR S..."
## [391,] "DELVE team memb..."        "G Hinton, R Nea..."
## [392,] "System and meth..."        "V Mnih, GE Hint..."
## [393,] "Deep learning..."          "G Hinton, Y LeC..."
## [394,] "Csc321. introdu..."        "G Hinton..."
## [395,] "Caching and rep..."        "S Becker, GE Hi..."
## [396,] "Embedding via c..."        "R Memisevic, G ..."
## [397,] "Learning distri..."        "A Paccanaro, GE..."
## [398,] "Pattern classif..."        "N Ueda, R Nakan..."
## [399,] "Fast neural net..."        "R Grzeszczuk, D..."
## [400,] "Parameter estim..."        "Z Ghahramani, G..."
## [401,] "Learning repres..."        "DE Rumelhart, G..."
## [402,] "Mental simulati..."        "G Hinton..."
## [403,] "Proceedings of ..."        "DS Touretzky, G..."
## [404,] "Learning intern..."        "DE Rumelhart, G..."
## [405,] "Parallel Distri..."        "DE Rumelhaxt, P..."
## [406,] "Parallel distri..."        "DE Rumelhart, G..."
## [407,] "Distributed rep..."        "GE Hinton, JL M..."
## [408,] "DE in, JL Rumal..."        "DE Rumelhart, G..."
## [409,] "Learning intern..."        "DE Rumelhart, G..."
## [410,] "Distributed rep..."        "GE Hinton..."
## [411,] "Parallel comput..."        "GE Hinton, P Sm..."
## [412,] "Parallel stocha..."        "TJ Sejnowski, G..."
## [413,] "Respectively Re..."        "G Hinton..."
## [414,] "Leaming represe..."        "DE Rumelhart, G..."
## [415,] "A comparison of..."        "T Schmah, G You..."
## [416,] "15 Learning to ..."        "GE Hinton, AD B..."
## [417,] "Nonlinear dimen..."        "R Salakhutdinov..."
## [418,] "Using mixtures ..."        "GE Hinton, M Re..."
## [419,] "Using neural ne..."        "GE Hinton, BJ F..."
## [420,] "The unity of co..."        "GE Hinton..."
## [421,] "Phoneme recogni..."        "WC Treurniet, M..."
## [422,] "Models of human..."        "GE Hinton..."
## [423,] "Parallel Distri..."        "DE Rumelhart, J..."
## [424,] "Parallel Distri..."        "DE Rumelhart, J..."
## [425,] "Parallel distri..."        "GE Hinton, TJ S..."
## [426,] "Technical Repor..."        "GE Hinton, TJ S..."
## [427,] "Learning Intern..."        "GE Hinton, RJ W..."
## [428,] "Hinton GE Willi..."        "D Rumelhart..."
## [429,] "Learning Intern..."        "DB Rumelhart, G..."
## [430,] "Who Said What: ..."        "MY Guan, V Guls..."
## [431,] "System and meth..."        "GE Hinton, A Kr..."
## [432,] "Fast Inference ..."        "N Srivastava, R..."
## [433,] "Deep generative..."        "GW Taylor, GE H..."
## [434,] "Modelling the s..."        "S Osindero, M W..."
## [435,] "A new learning ..."        "MW Hinton, M We..."
## [436,] "E cient stochas..."        "BJ Frey, GE Hin..."
## [437,] "Switching state..."        "Z Gahramani, GE..."
## [438,] "Using neural ne..."        "GE HINTON, P Da..."
## [439,] "Developing Popu..."        "RS Zemel, GE Hi..."
## [440,] "Theoretical psy..."        "A Kukla..."
## [441,] "SCENE-BASED AND..."        "LM Parsons, GE ..."
## [442,] "System and meth..."        "V Mnih, GE Hint..."
## [443,] "System and meth..."        "A Krizhevsky, I..."
## [444,] "System and meth..."        "A Krizhevsky, I..."
## [445,] "System and meth..."        "GE Hinton, A Kr..."
## [446,] "Package <U+0091>darch<U+0092>..."        "M Drees, J Ruec..."
## [447,] "Layer Normaliza..."        "J Lei Ba, JR Ki..."
## [448,] "System and meth..."        "A Krizhevsky, I..."
## [449,] "Training distil..."        "O Vinyals, JA D..."
## [450,] "time (min.)..."            "DE Rumelhart, G..."
## [451,] "Deterministic B..."        "GE Hinton..."
## [452,] "Updated Edition..."        "GE Hinton..."
## [453,] "System and meth..."        "A Krizhevsky, I..."
## [454,] "Dimensionality ..."        "M Sahani..."
## [455,] "Modeling Semant..."        "L Van Der Maate..."
## [456,] "Workshop summar..."        "K Yu, R Salakhu..."
## [457,] "Free Access A F..."        "GE Hinton..."
## [458,] "Free Access A F..."        "GE Hinton..."
## [459,] "Higher Educatio..."        "GE Hinton..."
## [460,] "Announcements o..."        "L Abbott, Y Ben..."
## [461,] "Copyright© 2006..."        "K Abbot-Smith, ..."
## [462,] "ROBOTICS: SCIEN..."        "C Breazeal, R F..."
## [463,] "15 Learning to ..."        "B Machine, GE H..."
## [464,] "Learning mixtur..."        "GE Hinton..."
## [465,] "Learning Popula..."        "GE Hinton..."
## [466,] "Department of C..."        "GE Hinton, Z Gh..."
## [467,] "Acharya, A., se..."        "GM Adelson-Vels..."
## [468,] "Learning fast n..."        "R Grzezczuk, D ..."
## [469,] "The DELVE user ..."        "CE Rasmussen, R..."
## [470,] "Glove-TalkII: M..."        "S Sidney Fels, ..."
## [471,] "Adults with bra..."        "GE Hinton, C Da..."
## [472,] "Book Review of ..."        "S Ahmad, T Sejn..."
## [473,] "Neural-net neig..."        "E Pennisi..."
## [474,] "Learning spatia..."        "S Becker, GE Hi..."
## [475,] "Connectionist M..."        "TJ Sejnowski, D..."
## [476,] "Artificial Neur..."        "GE Hinton, CKI ..."
## [477,] "The Development..."        "G Hinton, JL Mc..."
## [478,] "Connectionist M..."        "DS Touretzky, J..."
## [479,] "The development..."        "G HINTON, J MCC..."
## [480,] "Proceedings of ..."        "D Touretzky, G ..."
## [481,] "Connectionist M..."        "D Touretzky, G ..."
## [482,] "Speech recognit..."        "A Waibel, T Han..."
## [483,] "A New Approach ..."        "JL McClelland, ..."
## [484,] "KEVIN J. LANG A..."        "GE HINTON..."
## [485,] "Pittsburgh, PA ..."        "DC Plant, SJ No..."
## [486,] "Three frames su..."        "GE Hinton..."
## [487,] "Thanks to our g..."        "F Attneave, A B..."
## [488,] "Chapter IVb Som..."        "G Hinton..."
## [489,] "Technicalitepor..."        "GE Hinton, DH A..."
## [490,] "The editors gra..."        "NH Anderson, SM..."
## [491,] "HTA Whiting (ed..."        "G Hinton..."
## [492,] "Parallel Models..."        "MW Altom..."
## [493,] "Probabilistic r..."        "P DAYAN, GE HIN..."
## [494,] "Conditioned ref..."        "P DAYAN, GE HIN..."
## [495,] "Adaptive signal..."        "P DAYAN, GE HIN..."
## [496,] "Mean field theo..."        "P DAYAN, GE HIN..."
## [497,] "a Data-Glove an..."        "SS Fels, GE Hin..."
## [498,] "In M. Jenkin an..."        "BJ Frey, P Daya..."
## [499,] "Why the islands..."        "E Hutchiris, GE..."
## [500,] "Wackerly, DD, M..."        "DE Rumelhart, G..."
## [501,] "Making Connecti..."        "GE Hinton, SJ N..."
## [502,] "Massively paral..."        "SE Fahnn, GE Hi..."
## [503,] "John Hertz, And..."        "DH Ackley, GE H..."
## [504,] "eMMMMMS kkk kkk..."        "GE Hinton, DC P..."
## [505,] "imperfectly spe..."        "JL McCLELLAND, ..."
## [506,] "<U+7528><U+795E><U+7ECF><U+7F51><U+7EDC><U+51CF><U+5C11><U+6570><U+636E><U+7EF4><U+6570>..."
## [507,] "Sequential Thou..."        "GE HINTON..."
## [508,] "Aji, SM and RJ ..."        "K Abend, TJ Har..."
## [509,] "Learning Nonlin..."        "A Mnih, G Hinto..."
## [510,] "Archive for the..."        "LY Ku..."
## [511,] "Proceedings of ..."        "Connectionist M..."
## [512,] "Deep Neural Net..."        "G Hinton, L Den..."
## [513,] "Supplementary M..."        "H Larochelle, G..."
## [514,] "Supplemental Ma..."        "L van der Maate..."
## [515,] "User<U+0092>s Guide fo..."        "L van der Maate..."
## [516,] "Discovers Surfa..."        "GE Hinton, S Be..."
## [517,] "Supplementary M..."        "Y Tang, R Salak..."
## [518,] "JG CARBONELL..."           "S AMAREL, Y ANZ..."
## [519,] "Daniel G. BOBRO..."        "AG COHN, LC AIE..."
## [520,] "DANIEL G. BOBRO..."        "M BRADY, W BIBE..."
## [521,] "A. BUNDY..."               "LC AIELLO, S AM..."
## [522,] "Johnson-Laird, ..."        "RP Abelson, JB ..."
## [523,] "JG CARBONELL..."           "LC AIELLO, S AM..."
## [524,] "WJ CLANCEY inst..."        "MA AIZERMAN, S ..."
## [525,] "Daniel G. BOBRO..."        "LC AIELLO, S AM..."
## [526,] "R. DAVIS Al Lab..."        "R DECHTER, R MO..."
## [527,] "WJ CLANCEY..."             "MA AIZERMAN, S ..."
## [528,] "Stephen Grossbe..."        "N Suga, KJ Lang..."
## [529,] "R. DAVIS Al Lab..."        "AG COHN, LC AIE..."
## [530,] "GE Hinton Prefa..."        "GE Hinton, JB P..."
## [531,] "WJ CLANCEY..."             "S AMAREL, Y ANZ..."
## [532,] "BG BUCHANAN..."            "LC AIELLO, Y AN..."
## [533,] "JG CARBONELL Ca..."        "AG COHN, LC AIE..."
## [534,] "WJ CLANCEY..."             "LC AIELLO, S AM..."
## [535,] "Daniel G. BOBRO..."        "MA AIZERMAN, S ..."
## [536,] "AG COHN..."                "LC AIELLO, Y AN..."
## [537,] "JG GARBONELL..."           "AG COHN, LC AIE..."
## [538,] "Rarearch Notes ..."        "LC AIELLO, S AM..."
## [539,] "Associate Edito..."        "AG COHN, LC AIE..."
## [540,] "Ackley, DH, 147..."        "MA Arbib, W Bec..."
## [541,] "ImageNet Classi..."        "I Sutskever, GE..."
## [542,] "Admowledgement:..."        "J Allen, E Ande..."
## [543,] "E <U+0091>-Jgjfssl <U+0091>-y..."        "GE Hinton, DC P..."
## [544,] "THROUGH MATRIX ..."        "Y Le Cun, CC Ga..."
## [545,] "Inaugural Edito..."        "D Yu, G Hinton,..."
## [546,] "A Modi<U+FB01>ed Model..." "L Xul, MI Jorda..."
## [547,] "a Data-Glove an..."        "SS Fels, GE Hin..."
## [548,] "The Helmholtz M..."        "GEHP Dayaii, A ..."
## [549,] "A Bayesian Unsu..."        "Z Ghahramani, G..."
## [550,] "Local Physical ..."        "SOD Terzopoulos..."
## [551,] "Glove-TalkII: M..."        "G Hinton..."
## [552,] "Artificial Inte..."        "G Hinton..."
## [553,] "Fast Neural Net..."        "R Grzeszczuk, D..."
##        Journals             Year   Citations
##   [1,] "Nature 323, 533..." "1986" "34900"
##   [2,] "Parallel Distri..." "1986" "27417"
##   [3,] "CALIFORNIA UNIV..." "1985" "23094"
##   [4,] "MIT press 1, 18..." "1987" "18726"
##   [5,] "Advances in neu..." "2012" "15040"
##   [6,] "Neural computat..." "2006" "6618"
##   [7,] "MIT press..."       "1987" "6477"
##   [8,] "science 313 (57..." "2006" "5614"
##   [9,] "Nature 521 (755..." "2015" "3793"
##  [10,] "Neural computat..." "1991" "3551"
##  [11,] "Journal of mach..." "2014" "3515"
##  [12,] "Cognitive scien..." "1985" "3316"
##  [13,] "Journal of Mach..." "2008" "3149"
##  [14,] "IEEE Signal Pro..." "2012" "3147"
##  [15,] "Neural computat..." "2002" "2949"
##  [16,] "Learning in gra..." "1998" "2468"
##  [17,] "IEEE transactio..." "1989" "2396"
##  [18,] "arXiv preprint ..." "2012" "2111"
##  [19,] "Proceedings of ..." "2010" "2040"
##  [20,] "Artificial inte..." "1989" "1814"
##  [21,] "Parallel distri..." "1986" "1807"
##  [22,] "Parallel distri..." "1986" "1738"
##  [23,] "Technical repor..." "2009" "1546"
##  [24,] "Acoustics, spee..." "2013" "1437"
##  [25,] "Parallel distri..." "1986" "1385"
##  [26,] "Complex systems..." "1987" "1303"
##  [27,] "Lawrence Erlbau..." "1981" "1265"
##  [28,] "Advances in neu..." "2005" "1187"
##  [29,] "Artificial Inte..." "2009" "1074"
##  [30,] "Momentum, 1..."     "2010" "1017"
##  [31,] "Audio, Speech, ..." "2012" "1006"
##  [32,] "Proceedings of ..." "2007" "934"
##  [33,] "Parallel distri..." "1986" "899"
##  [34,] "Intel Technolog..." "2001" "883"
##  [35,] "Science 268 (52..." "1995" "868"
##  [36,] "Psychological r..." "1991" "852"
##  [37,] "Neural computat..." "1995" "811"
##  [38,] "Proceedings of ..." "1986" "811"
##  [39,] "Technical Repor..." "1996" "746"
##  [40,] "COURSERA: Neura..." "2012" "745"
## [41,] "Advances in neu..." "2003" "738"
## [42,] "International c..." "2013" "713"
## [43,] "International J..." "2009" "682"
## [44,] "Nature 323 (608..." "1986" "669"
## [45,] "Trends in cogni..." "2007" "646"
## [46,] "Carnegie-Mellon..." "1984" "633"
## [47,] "Neural networks..." "1990" "607"
## [48,] "Proceedings of ..." "1983" "577"
## [49,] "Advances in neu..." "2009" "552"
## [50,] "Neural networks..." "2012" "545"
## [51,] "Scientific Amer..." "1992" "545"
## [52,] "NeuroImage 16 (..." "2002" "537"
## [53,] "Advances in neu..." "2007" "499"
## [54,] "Technical Repor..." "1996" "481"
## [55,] "Proceedings of ..." "2011" "461"
## [56,] "arXiv preprint ..." "2015" "457"
## [57,] "Neural computat..." "1992" "456"
## [58,] "Parallel models..." "1981" "445"
## [59,] "Acoustics, Spee..." "2013" "442"
## [60,] "Aistats 10, 33-..." "2005" "442"
## [61,] "Proceedings of ..." "1993" "440"
## [62,] "..."                "1986" "430"
## [63,] "Advances in neu..." "1999" "426"
## [64,] "Nature 306 (593..." "1983" "421"
## [65,] "IEEE transactio..." "1993" "413"
## [66,] "Cognitive scien..." "1988" "413"
## [67,] "Advances in neu..." "1994" "405"
## [68,] "IEEE transactio..." "1997" "399"
## [69,] "Advances in neu..." "2005" "393"
## [70,] "Proceedings of ..." "2007" "388"
## [71,] "IET Digital Lib..." "1999" "368"
## [72,] "Advances in neu..." "1993" "359"
## [73,] "Nature 355 (635..." "1992" "355"
## [74,] "Advances in neu..." "2009" "348"
## [75,] "Proceedings of ..." "2010" "326"
## [76,] "International C..." "2007" "319"
## [77,] "Nips workshop o..." "2009" "301"
## [78,] "Artificial Inte..." "1990" "299"
## [79,] "Proceedings of ..." "1998" "291"
## [80,] "..."                "1983" "286"
## [81,] "Artificial Inte..." "1990" "282"
## [82,] "Advances in neu..." "1995" "281"
## [83,] "Neural computat..." "2000" "278"
## [84,] "Philosophical T..." "1997" "275"
## [85,] "Technical Repor..." "1988" "274"
## [86,] "Advances in neu..." "2009" "273"
## [87,] "The MIT Press..."   "1999" "269"
## [88,] "Acoustics, Spee..." "2013" "264"
## [89,] "Advances in neu..." "2009" "264"
## [90,] "IJCAI 85, 238-2..." "1985" "261"
## [91,] "Parallel models..." "1981" "261"
## [92,] "Advances in Neu..." "2015" "259"
## [93,] "Proceedings of ..." "1981" "256"
## [94,] "Proceedings of ..." "2009" "252"
##  [95,] "Acoustics, Spee..." "2013" "247"
##  [96,] "Computational N..." "2007" "247"
##  [97,] "Advances in neu..." "2002" "244"
##  [98,] "Journal of moto..." "1984" "232"
##  [99,] "Proceedings of ..." "2009" "229"
## [100,] "Advances in neu..." "2010" "222"
## [101,] "Advances in Neu..." "2009" "221"
## [102,] "Proceedings of ..." "1988" "219"
## [103,] "Acoustics, Spee..." "1988" "216"
## [104,] "IEEE Computer 2..." "1987" "214"
## [105,] "Computer Vision..." "2010" "212"
## [106,] "Cognitive Scien..." "1979" "212"
## [107,] "Acoustics, Spee..." "2011" "211"
## [108,] "Neural computat..." "1989" "203"
## [109,] "Proceedings of ..." "1981" "201"
## [110,] "Neural computat..." "2010" "197"
## [111,] "SIGIR Workshop ..." "2007" "194"
## [112,] "International C..." "1987" "192"
## [113,] "IEEE transactio..." "1998" "186"
## [114,] "International c..." "2010" "185"
## [115,] "Eleventh Annual..." "2010" "185"
## [116,] "IEEE transactio..." "1996" "185"
## [117,] "Acoustics, Spee..." "2012" "174"
## [118,] "European Confer..." "2010" "172"
## [119,] "JA Anderson and..." "1988" "171"
## [120,] "Artificial Inte..." "2007" "159"
## [121,] "ESANN..."           "2011" "157"
## [122,] "Advances in neu..." "2003" "153"
## [123,] "Neural computat..." "1998" "152"
## [124,] "Neural Networks..." "1996" "151"
## [125,] "Computer Vision..." "2011" "149"
## [126,] "arXiv preprint ..." "2015" "147"
## [127,] "Computer Vision..." "2007" "146"
## [128,] "Neural Computat..." "1997" "142"
## [129,] "Neural informat..." "1988" "142"
## [130,] "International C..." "2011" "141"
## [131,] "Advances in neu..." "1992" "138"
## [132,] "Perception 15 (..." "1986" "136"
## [133,] "Journal of Mach..." "2003" "135"
## [134,] "Computer Speech..." "1987" "134"
## [135,] "Physica D: Nonl..." "1986" "132"
## [136,] "Unpublished man..." "2010" "131"
## [137,] "Advances in neu..." "1995" "130"
## [138,] "Proceedings of ..." "1987" "129"
## [139,] "Advances in neu..." "2008" "128"
## [140,] "Scholarpedia 4 ..." "2009" "125"
## [141,] "Proceedings of ..." "1983" "125"
## [142,] "IEEE/ACM Transa..." "2014" "122"
## [143,] "Advances in neu..." "1991" "122"
## [144,] "Proceedings of ..." "1988" "122"
## [145,] "Attention and p..." "1981" "122"
## [146,] "Cognition 30 (1..." "1988" "121"
## [147,] "International C..." "2002" "120"
## [148,] "University of E..." "1978" "118"
## [149,] "arXiv preprint ..." "2016" "116"
## [150,] "Advances in Neu..." "2010" "113"
## [151,] "Advances in neu..." "2008" "112"
## [152,] "Philosophical T..." "2010" "111"
## [153,] "Computer Vision..." "2010" "109"
## [154,] "Neural Computat..." "2006" "109"
## [155,] "Advances in neu..." "2001" "107"
## [156,] "Journal of Mach..." "2004" "106"
## [157,] "IJCAI, 252-259..."  "1985" "103"
## [158,] "Acoustics, Spee..." "2011" "100"
## [159,] "Byte 10 (4), 26..." "1985" "99"
## [160,] "arXiv preprint ..." "2013" "97"
## [161,] "Proceedings of ..." "2012" "94"
## [162,] "Scientific Amer..." "1993" "93"
## [163,] "Technical Repor..." "1996" "91"
## [164,] "Computer Vision..." "2012" "90"
## [165,] "Vision, brain, ..." "1990" "89"
## [166,] "The Journal of ..." "2000" "84"
## [167,] "Proceedings of ..." "1995" "83"
## [168,] "Neural computat..." "2008" "81"
## [169,] "Biometrika 21, ..." "1967" "81"
## [170,] "Proc. Uncertain..." "2011" "80"
## [171,] "Neural Computat..." "1997" "80"
## [172,] "Advances in neu..." "1994" "77"
## [173,] "Statistics in m..." "1998" "76"
## [174,] "Advances in neu..." "2001" "75"
## [175,] "IEEE Transactio..." "1993" "75"
## [176,] "Journal of Mach..." "2011" "74"
## [177,] "Advances in neu..." "2009" "73"
## [178,] "Acoustics Speec..." "2010" "71"
## [179,] "Affective Compu..." "2008" "71"
## [180,] "Advances in Neu..." "1995" "70"
## [181,] "IET Digital Lib..." "1997" "69"
## [182,] "Topics in Cogni..." "2011" "68"
## [183,] "NIPS 2858, 2859..." "2010" "67"
## [184,] "Advances in neu..." "2006" "66"
## [185,] "Frontiers in Ha..." "2004" "66"
## [186,] "Advances in Neu..." "2012" "65"
## [187,] "Acoustics, Spee..." "2011" "64"
## [188,] "Advances in neu..." "2009" "63"
## [189,] "Parallel distri..." "6"    "62"
## [190,] "Artificial Inte..." "2007" "61"
## [191,] "Proc. ICML Work..." "2013" "60"
## [192,] "MIT Press, Camb..." "1986" "60"
## [193,] "Proceedings of ..." "1990" "59"
## [194,] "Coursera, video..." "2012" "58"
## [195,] "Graphics Interf..." "2002" "58"
## [196,] "Machine learnin..." "2012" "57"
## [197,] "University of T..." "2002" "56"
## [198,] "Neural Computat..." "1993" "55"
## [199,] "MIT Press, Camb..." "1986" "54"
## [200,] "Computer Vision..." "2011" "53"
## [201,] "Advances in neu..." "1996" "53"
## [202,] "Connectionist m..." "1991" "53"
## [203,] "Encyclopedia of..." "2011" "52"
## [204,] "Cognitive scien..." "2006" "52"
## [205,] "IEEE Transactio..." "2001" "51"
## [206,] "Neural Computat..." "1999" "50"
## [207,] "NATO ASI SERIES..." "1998" "49"
## [208,] "Advances in neu..." "2010" "48"
## [209,] "URL http://www...." "1996" "48"
## [210,] "AIP conference ..." "1986" "48"
## [211,] "Parallel distri..." ""     "47"
## [212,] "Advances in Neu..." "2016" "45"
## [213,] "Advances in Neu..." "1992" "41"
## [214,] "Advances in neu..." "1991" "41"
## [215,] "Neural Computat..." "1990" "41"
## [216,] "World Congress ..." "1994" "40"
## [217,] "Human-Computer ..." "1984" "40"
## [218,] "Backpropagation..." "1995" "38"
## [219,] "IEEE Computer S..." "1985" "38"
## [220,] "Cognitive scien..." "2014" "37"
## [221,] "AISTATS..."         "2001" "37"
## [222,] "Advances in neu..." "2000" "37"
## [223,] "Advances in neu..." "2003" "36"
## [224,] "Human Motor Act..." "1984" "35"
## [225,] "Neural computat..." "2010" "34"
## [226,] "IJCAI 5, 1765-1..." "2005" "34"
## [227,] "Advances in neu..." "2000" "34"
## [228,] "San Francisco, ..." "1994" "34"
## [229,] "arXiv preprint ..." "2012" "33"
## [230,] "Advances in neu..." "2005" "33"
## [231,] "URL http://www...." "1996" "33"
## [232,] "Advances in neu..." "1990" "32"
## [233,] "IEEE transactio..." "2013" "31"
## [234,] "Computer Graphi..." "2002" "30"
## [235,] "Proceedings of ..." "2001" "30"
## [236,] "arXiv preprint ..." "2012" "29"
## [237,] "Solid-State Cir..." "2001" "29"
## [238,] "Advances in neu..." "1998" "29"
## [239,] "DE Rumelhart, J..." "1987" "29"
## [240,] "arXiv preprint ..." "2017" "28"
## [241,] "International C..." "2013" "27"
## [242,] "Advances in Neu..." "2001" "27"
## [243,] "..."                "2009" "26"
## [244,] "University of T..." "1994" "26"
## [245,] "Advances in neu..." "1990" "26"
## [246,] "MIT Press, Camb..." "1986" "26"
## [247,] "AISTATS..."         "2005" "25"
## [248,] "Perception 13 (..." "1984" "25"
## [249,] "Coursera Lectur..." "2012" "24"
## [250,] "Psychological I..." "1995" "24"
## [251,] "Chicago: Psycho..." "1986" "24"
## [252,] "Neural Networks..." "2010" "23"
## [253,] "Int. Conf. on I..." "2001" "23"
## [254,] "Proceedings of ..." "1990" "23"
## [255,] "IEEE Transactio..." "2012" "22"
## [256,] "ICANN<U+0092>93, 11-18..." "1993" "22"
## [257,] "MIT Press 1, 31..." "1986" "21"
## [258,] "Neural networks..." "1996" "20"
## [259,] "D. Rumelhart an..." "1987" "20"
## [260,] "Behavioral and ..." "1979" "20"
## [261,] "30th Internatio..." "2013" "19"
## [262,] "Canadian Psycho..." "2003" "19"
## [263,] "Statistics and ..." "1998" "19"
## [264,] "Computational a..." "1997" "19"
## [265,] "Morgan Kaufmann..." "2014" "18"
## [266,] "Fifteenth Annua..." "2014" "18"
## [267,] "International C..." "2008" "18"
## [268,] "Adv. Neural Inf..." "2004" "18"
## [269,] "Connectionist M..." "1991" "18"
## [270,] "Advances in neu..." "1989" "18"
## [271,] "NIPS Workshop..."   "2009" "17"
## [272,] "Computer vision..." "1997" "17"
## [273,] "Proc. of the AI..." "1976" "17"
## [274,] "INTERSPEECH, 17..." "2013" "16"
## [275,] "Technical Repor..." "2003" "16"
## [276,] "Parallel distri..." "1986" "16"
## [277,] "The MIT Press, ..." ""     "16"
## [278,] "Neurocomputing ..." "2009" "15"
## [279,] "Artificial Inte..." "2007" "15"
## [280,] "Advances in neu..." "2002" "15"
## [281,] "Advances in Neu..." "1996" "15"
## [282,] "MIT Press, Camb..." "1986" "15"
## [283,] "Third Annual Co..." "1981" "15"
## [284,] "Encyclopedia of..." "2011" "14"
## [285,] "Proceedings of ..." "2003" "14"
## [286,] "Neural Networks..." "2000" "14"
## [287,] "Le débat, 45-64..." "1987" "14"
## [288,] "MIT Press, Camb..." "1986" "14"
## [289,] "MIT Press, Camb..." "1986" "14"
## [290,] "MIT Press, Camb..." "1986" "14"
## [291,] "MIT Press, Camb..." "1986" "14"
## [292,] "Proceedings of ..." "2009" "13"
## [293,] "Advances in Neu..." "2009" "13"
## [294,] "Neural networks..." "2005" "13"
## [295,] "Dordrecht: Kluw..." "1998" "13"
## [296,] "The Computer Jo..." "1997" "13"
## [297,] "Parallel Distri..." "1985" "13"
## [298,] "Behavioral and ..." "1980" "13"
## [299,] "ESANN..."           "2009" "12"
## [300,] "..."                "1992" "12"
## [301,] "Proc. 1988 Conn..." "1988" "12"
## [302,] "arXiv preprint ..." "2017" "11"
## [303,] "Communications ..." "2011" "11"
## [304,] "Invited talk at..." "2007" "11"
## [305,] "Cognitive Model..." "2002" "11"
## [306,] "Machine Learnin..." "1990" "11"
## [307,] "..."                "2003" "10"
## [308,] "Preprint..."        "1997" "10"
## [309,] "ICANN-95, 483-4..." "1995" "10"
## [310,] "Cybernetics and..." "1994" "10"
## [311,] "..."                "1993" "10"
## [312,] "MIT Press, Camb..." "1986" "10"
## [313,] "MIT Press, Camb..." ""     "10"
## [314,] "Advances in Neu..." "2011" "9"
## [315,] "nature neurosci..." "2000" "9"
## [316,] "Department of C..." "1996" "9"
## [317,] "Spatial Vision ..." "1993" "9"
## [318,] "Advances in neu..." "1990" "9"
## [319,] "Morgan Kaufmann..." "1990" "9"
## [320,] "Genetic Algorit..." "1987" "9"
## [321,] "MIT Press, Camb..." "1986" "9"
## [322,] "Cambridge, MA: ..." "1986" "9"
## [323,] "I, chapter Lear..." "1986" "9"
## [324,] "IBM Hursley Hum..." "1983" "9"
## [325,] "International J..." "2015" "8"
## [326,] "Neural Networks..." "2005" "8"
## [327,] "Data Compressio..." "1996" "8"
## [328,] "Advances in neu..." "1995" "8"
## [329,] "IEEE transactio..." "1989" "8"
## [330,] "Cambridge, MA, ..." "1986" "8"
## [331,] "MIT Press, Camb..." "1985" "8"
## [332,] "DTIC Docum..."      ""     "8"
## [333,] "IEEE Transactio..." ""     "8"
## [334,] "Neural systems ..." "2011" "7"
## [335,] "IEEE transactio..." "2004" "7"
## [336,] "Advances in Neu..." "1992" "7"
## [337,] "Proceedings of ..." "1986" "7"
## [338,] "MIT Press..."       "1986" "7"
## [339,] "Parallel distri..." "1986" "7"
## [340,] "Proceedings of ..." "1978" "7"
## [341,] "A learning algo..." ""     "7"
## [342,] "Advances In Neu..." "2016" "6"
## [343,] "Presented as th..." "2014" "6"
## [344,] "arXiv preprint ..." "2013" "6"
## [345,] "WISP-2001 Works..." "2001" "6"
## [346,] "AAAI/IAAI, 1159..." "2000" "6"
## [347,] "ICML, 711-718..."   "2000" "6"
## [348,] "..."                "1996" "6"
## [349,] "Introducción al..." "1992" "6"
## [350,] "Art. Neural Sys..." "1992" "6"
## [351,] "Abstracts 1st I..." "1988" "6"
## [352,] "Perception 16 (..." "1987" "6"
## [353,] "Japan: Advanced..." "1987" "6"
## [354,] "The MIT Press, ..." "1986" "6"
## [355,] "Parallel Distri..." ""     "6"
## [356,] "MIT Press: Camb..." ""     "6"
## [357,] "arXiv preprint ..." "2013" "5"
## [358,] "..."                "2012" "5"
## [359,] "JPHBS 3, 74-8..."   "2012" "5"
## [360,] "ESANN, 493-498..."  "2008" "5"
## [361,] "Canada, Tech. R..." "1997" "5"
## [362,] "Proceedings of ..." "1996" "5"
## [363,] "..."                "1991" "5"
## [364,] "American Associ..." "1988" "5"
## [365,] "MIT Press, Camb..." "1986" "5"
## [366,] "Cambridge, MIT ..." "1986" "5"
## [367,] "MIT Press..."       "1986" "5"
## [368,] "Nature, London..."  "1986" "5"
## [369,] "Cambridge, MA: ..." "1986" "5"
## [370,] "Paralled distri..." "1986" "5"
## [371,] "MIT Press..."       "1986" "5"
## [372,] "MIT Press..."       "1986" "5"
## [373,] "MIT Press Cambr..." "1986" "5"
## [374,] "MIT Press..."       "1986" "5"
## [375,] "Nature 23, 533-..." ""     "5"
## [376,] "Parallel distri..." ""     "5"
## [377,] "DE Rumelhart, J..." ""     "5"
## [378,] "URL http://www...." ""     "5"
## [379,] "..."                ""     "5"
## [380,] "..."                "2010" "4"
## [381,] "Google Tech Tal..." "2007" "4"
## [382,] "..."                "2007" "4"
## [383,] "Advances in Neu..." "2002" "4"
## [384,] "IET Digital Lib..." "1999" "4"
## [385,] "Kluwer, Academi..." "1998" "4"
## [386,] "Network: Comput..." "1998" "4"
## [387,] "Parallel Distri..." "1988" "4"
## [388,] "MIT Press..."       "1986" "4"
## [389,] "Parallel distri..." "1986" "4"
## [390,] "SCIENCE, www. s..." ""     "4"
## [391,] "Technical repor..." ""     "4"
## [392,] "US Patent 9,704..." "2017" "3"
## [393,] "Nature 521, 436..." "2015" "3"
## [394,] "Lecture 10..."      "2010" "3"
## [395,] "Cosyne (abstrac..." "2007" "3"
## [396,] "Neural Networks..." "2005" "3"
## [397,] "Neural Nets WIR..." "2002" "3"
## [398,] "Neural Networks..." "1999" "3"
## [399,] "Advances in neu..." "1999" "3"
## [400,] "CRG-TR-96-2..."     "1996" "3"
## [401,] "Spie Milestone ..." "1994" "3"
## [402,] "Nature 347 (629..." "1990" "3"
## [403,] "Morgan Kaufmann..." "1989" "3"
## [404,] "MIT Press, Camb..." "1986" "3"
## [405,] "Cambridge: The ..." "1986" "3"
## [406,] "Cambridge: MIT ..." "1986" "3"
## [407,] "MIT Press, Camb..." "1986" "3"
## [408,] "MIT Press, Camb..." "1986" "3"
## [409,] "MIT Press, Camb..." "1986" "3"
## [410,] "cambridge, ma: ..." "1986" "3"
## [411,] "Cognitive Scien..." "1984" "3"
## [412,] "Johns Hopkins U..." "1984" "3"
## [413,] "Pragmatics Micr..." "1978" "3"
## [414,] "Natgre 32 (3), ..." ""     "3"
## [415,] "Neuroimage 47, ..." "2009" "2"
## [416,] "Probabilistic M..." "2002" "2"
## [417,] "RBM 2, 1000..."     "2000" "2"
## [418,] "Unpublished..."     "1998" "2"
## [419,] "MECHANICAL WORK..." "1996" "2"
## [420,] "Memories, Thoug..." "1991" "2"
## [421,] "Neural Networks..." "1988" "2"
## [422,] "Computational i..." "1987" "2"
## [423,] "MIT Press. Camb..." "1987" "2"
## [424,] "The MIT Press, ..." "1986" "2"
## [425,] "Cambridge, MA: ..." "1986" "2"
## [426,] "Carnegie Mellon..." "1984" "2"
## [427,] "Parallel Distri..." ""     "2"
## [428,] "MIT pr..."          ""     "2"
## [429,] "Cambridge. MA: ..." ""     "2"
## [430,] "arXiv preprint ..." "2017" "1"
## [431,] "US Patent 9,406..." "2016" "1"
## [432,] "..."                "2013" "1"
## [433,] "Proc. 4th Int. ..." "2008" "1"
## [434,] "Neural Computat..." "2006" "1"
## [435,] "..."                "2002" "1"
## [436,] "Computer Journa..." "1997" "1"
## [437,] "Department of C..." "1996" "1"
## [438,] "AMER STATISTICA..." "1994" "1"
## [439,] "AAAI Fall Sympo..." "1993" "1"
## [440,] "American Psycho..." "1990" "1"
## [441,] "BULLETIN OF THE..." "1988" "1"
## [442,] "US Patent App. ..." "2017" "0"
## [443,] "US Patent 9,563..." "2017" "0"
## [444,] "US Patent App. ..." "2017" "0"
## [445,] "US Patent App. ..." "2016" "0"
## [446,] "..."                "2016" "0"
## [447,] "arXiv preprint ..." "2016" "0"
## [448,] "US Patent 9,251..." "2016" "0"
## [449,] "US Patent App. ..." "2015" "0"
## [450,] "Advanced Contro..." "2014" "0"
## [451,] "Connectionist M..." "2014" "0"
## [452,] "Parallel Models..." "2014" "0"
## [453,] "US Patent App. ..." "2013" "0"
## [454,] "..."                "2011" "0"
## [455,] "..."                "2009" "0"
## [456,] "Proceedings of ..." "2009" "0"
## [457,] "Neural Computat..." "2007" "0"
## [458,] "Neural Computat..." "2006" "0"
## [459,] "Science 313, 50..." "2006" "0"
## [460,] "Neurocomputing ..." "2006" "0"
## [461,] "Cognitive Scien..." "2006" "0"
## [462,] "..."                "2005" "0"
## [463,] "Probabilistic M..." "2002" "0"
## [464,] "Unsupervised Le..." "1999" "0"
## [465,] "Unsupervised Le..." "1999" "0"
## [466,] "..."                "1997" "0"
## [467,] "Artificial Inte..." "1997" "0"
## [468,] "ACM SIGGRAPH 97..." "1997" "0"
## [469,] "..."                "1996" "0"
## [470,] "ADVANCES IN NEU..." "1995" "0"
## [471,] "SCIENTIFIC AMER..." "1993" "0"
## [472,] "Artificial Inte..." "1993" "0"
## [473,] "Science News 14..." "1992" "0"
## [474,] "SPIE Int. Conf...." "1991" "0"
## [475,] "Morgan Kaufmann..." "1991" "0"
## [476,] "Proceedings of ..." "1991" "0"
## [477,] "CARNEGIE-MELLON..." "1990" "0"
## [478,] "CALIFORNIA UNIV..." "1990" "0"
## [479,] "..."                "1990" "0"
## [480,] "CARNEGIE-MELLON..." "1989" "0"
## [481,] "Morgan Kaufmann..." "1989" "0"
## [482,] "The Journal of ..." "1988" "0"
## [483,] "Le Débat, 45-64..." "1987" "0"
## [484,] "Neural networks..." "1987" "0"
## [485,] "..."                "1986" "0"
## [486,] "Behavioral and ..." "1985" "0"
## [487,] "Cognition 19, 2..." "1985" "0"
## [488,] "Advances in Psy..." "1984" "0"
## [489,] "..."                "1984" "0"
## [490,] "Acta Psychologi..." "1984" "0"
## [491,] "Human Motor Act..." "1983" "0"
## [492,] "American Scient..." "1983" "0"
## [493,] "..."                ""     "0"
## [494,] "..."                ""     "0"
## [495,] "..."                ""     "0"
## [496,] "..."                ""     "0"
## [497,] "..."                ""     "0"
## [498,] "..."                ""     "0"
## [499,] "..."                ""     "0"
## [500,] "..."                ""     "0"
## [501,] "Evolutionary Ps..." ""     "0"
## [502,] "..."                ""     "0"
## [503,] "Learning 1 (312..." ""     "0"
## [504,] "..."                ""     "0"
## [505,] "..."                ""     "0"
## [506,] "..."                ""     "0"
## [507,] "..."                ""     "0"
## [508,] "Science 9, 147-..." ""     "0"
## [509,] "..."                ""     "0"
## [510,] "..."                ""     "0"
## [511,] "Morgan Kaufmann..." ""     "0"
## [512,] "..."                ""     "0"
## [513,] "..."                ""     "0"
## [514,] "..."                ""     "0"
## [515,] "..."                ""     "0"
## [516,] "..."                ""     "0"
## [517,] "..."                ""     "0"
## [518,] "..."                ""     "0"
## [519,] "..."                ""     "0"
## [520,] "..."                ""     "0"
## [521,] "..."                ""     "0"
## [522,] "..."                ""     "0"
## [523,] "..."                ""     "0"
## [524,] "..."                ""     "0"
## [525,] "..."                ""     "0"
## [526,] "..."                ""     "0"
## [527,] "..."                ""     "0"
## [528,] "..."                ""     "0"
## [529,] "..."                ""     "0"
## [530,] "..."                ""     "0"
## [531,] "..."                ""     "0"
## [532,] "..."                ""     "0"
## [533,] "..."                ""     "0"
## [534,] "..."                ""     "0"
## [535,] "..."                ""     "0"
## [536,] "..."                ""     "0"
## [537,] "..."                ""     "0"
## [538,] "..."                ""     "0"
## [539,] "..."                ""     "0"
## [540,] "..."                ""     "0"
## [541,] "..."                ""     "0"
## [542,] "..."                ""     "0"
## [543,] "..."                ""     "0"
## [544,] "..."                ""     "0"
## [545,] "..."                ""     "0"
## [546,] "..."                ""     "0"
## [547,] "..."                ""     "0"
## [548,] "..."                ""     "0"
## [549,] "..."                ""     "0"
## [550,] "..."                ""     "0"
## [551,] "..."                ""     "0"
## [552,] "Van Nostrand's ..." ""     "0"
## [553,] "..."                ""     "0"
```

Bibliography

1. http://people.cs.ksu.edu/ schmidt/115/ch7.html, "Chapter 7: Data Compression", visited 09/14/2017
2. https://en.wikipedia.org/wiki/Data_deduplication, "Data deduplication", visited 09/14/2017

Appendix: Full Codes

Problem 2

a) Google Scholar citation function: ID and html

```r
# Function P2(a)
CitesScholar <- function(name) {
  # Turning off the warnings for visualization purposes
  options(warn = -1)

  # Loading libraries
  suppressMessages(library(XML))
  suppressMessages(library(RCurl))
  suppressMessages(library(stringr))

  # Check if the input is a string
  if (!is.character(name)){
    stop("The input must be a name (e.g Geoffrey Hinton)")
  }

  # Check if we have only letters and a first and last name
  if (grepl("^[[:alpha:]]+[[:space:]]+[[:alpha:]]+$", name) == FALSE){
    stop(paste("Only two words (no digits) are allowed as the author's name.",
               "The name string must have a space between the first and last names",
               "please try again (e.g Geoffrey Hinton)", sep = " "))
  }

  # Separate the name in first and last names
  firstname <- strsplit(name, " ")[[1]][[1]]
  lastname <- strsplit(name, " ")[[1]][[2]]

  # Generate the URL with the profile in google scholar
  urlInit <- paste("https://scholar.google.com/scholar?hl=es&q=", firstname,
                   "+", lastname, "&btnG=&lr=", sep = "")

  # Get the Google Scholar ID and the link to the personal citations website
  rawHTML <- readLines(urlInit)

  # Parse the raw HTML code
  parsedHTML <- htmlParse(rawHTML)

  # Find the link to the author's personal website
  link <- xpathSApply(parsedHTML, "//h4[@class = 'gs_rt2']//a",
                      xmlGetAttr, "href")
  # Check if there are valid links inside the file: if not, stop
  if (length(link) <= 0) {
    stop(paste("The author does not exist in the Google Scholar Database,",
               "please try again with a different name", sep = " "))
  }

  # Extract the user
  UserID <- str_match(link, ".*user=([\\d\\w]+)&")[[2]]

  # If no User ID, then author does not exists in Google Scholar: stop
  if (is.na(UserID) == TRUE){
    stop(paste("The author does not exist in the Google Scholar Database,",
               "please try again with a different name", sep = " "))
  }

  # Get the citations html
  urlcites <- paste("https://scholar.google.com", link, sep = "")
  htmlcites <- readLines(urlcites)
  parhtmlcites <- htmlParse(htmlcites)

  # Return the user ID and parsed html from the personal citations web
  returnlist <- list(UserID, parhtmlcites)
  return(returnlist)
}
```

Run the code

```
# Obtain the User ID and HTML object of the author
ResultsList <- CitesScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Display html class
class(Authorhtml)[[1]]

# Display outputs (summary of the html file for visualization purposes)
# User ID
UserID

# HTML summary
summary(Authorhtml)
```

a.2) Extra function

```r
DownloadfromScholar <- function(name) {
  # Loading libraries
  library(XML)
  library(stringr)

  # Turning off the warnings for visualization purposes
  options(warn = -1)

  # Check if the input is a string
  if (!is.character(name)){
    stop("The input must be a name (e.g Geoffrey Hinton)")
  }

  # Check if we have only letters and a first and last name
  if (grepl("^[[:alpha:]]+[[:space:]]+[[:alpha:]]+$", name) == FALSE){
    stop(paste("Only two words (no digits) are allowed as the author's name,",
               "please try again (e.g Geoffrey Hinton)", sep = " "))
  }

  # Check if the name string has a first and last name
  if (length(strsplit(name, " ")[[1]]) < 2){
    stop(paste("The name string must have a space between the first and last names,",
               "please try again (e.g Geoffrey Hinton)", sep = " "))
  }

  # Separate the name in first and last names
  firstname <- strsplit(name, " ")[[1]][[1]]
  lastname <- strsplit(name, " ")[[1]][[2]]

  # Generate the initial URL to download the .html file
  urlInit <- paste("https://scholar.google.com/scholar?q=", firstname,
                   "+", lastname, "&hl=en&as_sdt=0,5", sep = "")
  filename <- paste(firstname, lastname, ".html", sep = "")

  # Download the file and parse the html file
  download.file(urlInit, filename)
  parsedHTML <- htmlParse(readLines(filename))

  # Find the link to the author's personal website
  link <- xpathSApply(parsedHTML, "//h4[@class = 'gs_rt2']//a",
                      xmlGetAttr, "href")

  # Check if the link exists (is valid)
  if (length(link) <= 0) {
    stop(paste("The author does not exist in the Google Scholar Database,",
               "please try again with a different name", sep = " "))
  }

  # Extract the user
  UserID <- str_match(link, ".*user=([\\d\\w]+)&")[[2]]
  # If no User ID, then author does not exists in Google Scholar: stop
  if (is.na(UserID) == TRUE){
    stop(paste("The author does not exist in the Google Scholar Database,",
               "please try again with a different name", sep = " "))
  }

  # Get the citations html: generate the url and download it
  urlcites <- paste("https://scholar.google.com", link, sep = "")
  filenamecites <- paste(UserID, ".html", sep = "")
  download.file(urlcites, filenamecites)

  # Parse the new html
  parhtmlcites <- htmlParse(readLines(filenamecites))

  # Return the user ID and parsed html from the personal citations web
  returnlist <- list(UserID, parhtmlcites)
  return(returnlist)
}
```

Run the code

```
# Obtain the User ID and HTML object of the author
ResultsList <- DownloadfromScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Display html class
class(Authorhtml)[[1]]

# Display them (summary of the html file for visualization purposes)
# User ID
UserID

# HTML summary
summary(Authorhtml)
```

b) First 20 citations as DataFrame

```r
# Function definition
CreateDataFrame_subset <- function(html){
  # Turning off the warnings for visualization purposes
  options(warn = -1)

  # Libraries
  suppressMessages(library(XML))
  suppressMessages(library(RCurl))
  suppressMessages(library(stringr))

  # Check if the file is a parsed HTML
  if (typeof(html) != "externalptr"){
    stop("Not a valid html file from author's citation website")
  }

  if (class(html)[[1]] != "HTMLInternalDocument") {
    stop(paste("Not a valid html file from author's citation website",
               "please check the format of the input file", sep = " "))
  }

  # Articles titles
  Titles <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//tr[@class ='gsc_a_tr']
                        //td[@class = 'gsc_a_t']//a[@class = 'gsc_a_at']",
                        xmlValue, trim = TRUE, encoding = "UTF-8")

  # Check if there are valid titles: if not, not a valid file
  if (length(Titles) <= 0){
    stop(paste("There is no author's citation page inside Google Scholar",
               "associated with the html file being processed, please",
               "try again with a different file", sep = " "))
  }

  # Journals and authors array
  JournalAuthors <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                                tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_t']
                                //div[@class = 'gs_gray']", xmlValue,
                                trim = TRUE, encoding = "UTF-8")

  # Getting the Journals (even numbers of previos vector)
  Journals <- JournalAuthors[seq(2, length(JournalAuthors), 2)]

  # We delete the year of publication after the Journal
  Journals <- substring(Journals, 0, nchar(Journals) - 6)

  # Getting the authors (odd numbers)
  Authors <- JournalAuthors[seq(1, length(JournalAuthors), 2)]

  # Years of publication
  Year <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//tr[@class ='gsc_a_tr']
                      //td[@class = 'gsc_a_y']//span[@class = 'gsc_a_h']",
                      xmlValue, trim = TRUE, encoding = "UTF-8")
  # Number of citations (including repeated articles)
  Citations <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']
                         //tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_c']
                         //a", xmlValue, trim = TRUE, encoding = "UTF-8")

  # Delete potential non numerical elements by transforming and filtering the array
  Citations <- as.numeric(Citations)
  remove <- c(NA)

  # Re-Transform it to a character array for the DF
  Citations <- as.character(Citations[!Citations %in% remove])

  # Dimension checks: fill with 0 empty citations
  if (length(Citations) != length(Titles)){
    Citations <- c(Citations, integer(length(Titles) - length(Citations)))
  }
  # Create a DataFrame
  x_name <- "Title"
  y_name <- "Authors"
  z_name <- "Journals"
  w_name <- "Year"
  v_name <- "Citations"

  # Create the DF using the reshape2 melt function for simplicity
  suppressMessages(require(reshape2))
  suppressMessages(ScholarDF <- melt(data.frame(Titles, Authors, Journals,
                                              Year, Citations)))

  # Define the columns' names
  colnames(ScholarDF) <- c(x_name, y_name, z_name, w_name, v_name)

  # Return the DataFrame
  return(ScholarDF)
}
```

Run the code

```r
# Call the function and create a DataFrame from the author's website
ResultsList <- CitesScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Create the data frame
SubsetScholarDF <- CreateDataFrame_subset(Authorhtml)

# Check the length of any column
length(SubsetScholarDF$Title)

# Printing a summary using the apply function (``pretty'' output)
apply(SubsetScholarDF, 2, function(x)
  if (length(x) > 15 && is.na(str_match(x, "^[[:digit:]]"))) {
    paste(substr(x, start = 0, stop = 15), "...", sep = "")
  }
  else {substr(x, start = 0, stop = length(x))
  }
)
```

c) Testing our code

c.1) CitesScholar Tests

```r
setwd("C:/Users/chile/ps2/RTests/")
source("CitesScholar.R")

# Testing the CitesScholar function (input = HTML, output = list)
test_that("Types and lengths are consistent when using a real author's name", {
  # As an example, we are using the citation website of the well known Albert Einstein
  ResultsList <- CitesScholar("Andres Weintraub")

  # Check if the final output is a list
  expect_that( ResultsList, is_a("list") )

  # Check its length
  expect_that( length(ResultsList), equals(2) )

  # Check the types of both outputs
  expect_that( ResultsList[[1]], is_a("character") )
  expect_that( ResultsList[[2]], is_a("HTMLInternalDocument") )
})

# Testing the CitesScholar function (input = HTML, output = list)
test_that("Output when a non-existent author name is provided", {
  # A non-registered user name is provided
  ResultsList <- CitesScholar("Cristobal Pais")
})

# Testing the CitesScholar function for different inputs
test_that("Only first name", {
  # One word is provided
  ResultsList <- CitesScholar("Albert")
})

test_that("More than two names", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("Albert Einstein Canalejo")
})

test_that("Words and numbers", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("Albert Einstein2")
})

test_that("Words and numbers (mixed)", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("Albert E1nstein")
})

test_that("No space between names", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("AlbertE1nstein")
})
```

```r
test_that("Uppercase name", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("ALBERT EINSTEIN")
})

test_that("Lowercase name", {
  # More than 2 words are provided
  ResultsList <- CitesScholar("albert einstein")
})

test_that("Erroneous input is provided (numbers)", {
  # Numeric
  ResultsList <- CitesScholar(1123)
})

test_that("Erroneous input is provided (digit strings)", {
  # String with digits
  FinalDF2 <- CitesScholar("1222")
})

test_that("Erroneous input is provided (anything)", {
  # Html sintaxis
  FinalDF <- CitesScholar("<html><body>")
})
```

Run the code

```r
# Loading the library
library(testthat)

# Set working directory
setwd("C:/Users/chile/ps2/RTests/")

# Invoking the test
test_file("test_CitesScholar.R")
```

c.2) CreateDataFrame_subset Tests

```r
setwd("C:/Users/chile/ps2/RTests/")
source("CreateDataFrame_subset.R")

# Testing the CreateDataFrame_subset function (input = HTML, output = Dataframe)
test_that("Types and lengths are consistent when using a known html file", {
  # Save the output in the FinalDF variable, reading a html file from Google Scholar
  # As an example, we are using the citation website of Albert Einstein
  FinalDF <- CreateDataFrame_subset(htmlParse(readLines("qc6CJjYAAAAJ.html")))

  # Check if the final output is a DataFrame
  expect_that( FinalDF, is_a("data.frame") )

  # Check if all columns have the same length
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Authors)) )
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Year)) )
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Citations)) )
  expect_that( length(FinalDF$Title), equals(length(FinalDF$Journals)) )

  # Check if the Citations column has only numbers
  expect_that( as.numeric(FinalDF$Citations), is_a("numeric") )
})

# Testing the CreateDataFrame_subset function for different inputs
test_that("Output when a non-existent user html file is provided", {
  # Save the output in the FinalDF variable, after reading a html
  # file from Google Scholar website, but from a non-existing user
  FinalDF2 <- CreateDataFrame_subset(htmlParse(readLines("nonexistent.html")))
})

# Another html file (not from Google Scholar)
test_that("A html file/code from any page", {
  # Html taken from google.com
  FinalDF <- CreateDataFrame_subset(htmlParse(readLines("index.html")))
})

# Numbers as inputs
test_that("Erroneous input is provided (numbers)", {
  # Testing with a numeric input
  FinalDF2 <- CreateDataFrame_subset(1222)
})

# String with numbers
test_that("Erroneous input is provided (digit strings)", {
  # String composed by digits
  FinalDF2 <- CreateDataFrame_subset("1222")
})

# Any random string
test_that("Erroneous input is provided (character strings)", {
  # Characters string
  FinalDF2 <- CreateDataFrame_subset("hello")
})

# Some html sintax (check if function is confused)
test_that("Erroneous input is provided (html sintax)", {
  # Html sintaxis
  FinalDF <- CreateDataFrame_subset("<html><body>")
})
```

Run the code

```r
# Loading the library
library(testthat)

# Set working directory
setwd("C:/Users/chile/ps2/RTests/")

# Downloading the files (using wget for simplicity): valid, non-existent and non-valid
system('bash -c "wget -q -O qc6CJjYAAAAJ.html https://scholar.google.com/\
       citations?user=qc6CJjYAAAAJ&hl=es&oi=ao"')
system('bash -c "wget -q -O nonexistent.html https://scholar.google.com/\
       \scholar?q=Cristobal+Pais&btnG=&hl=es&as_sdt=0%2C5"')
system('bash -c "wget -q www.google.com"')

# Invoking the test
test_file("test_CreateDataFrame_subset.R")
```

d) Extra Credit: All citations as DataFrame

```r
# Declaring the dunction
CreateDataFrame <- function(html){
  # Turning off the warnings for visualization purposes
  options(warn = -1)

  # Libraries
  suppressMessages(library(XML))
  suppressMessages(library(RCurl))
  suppressMessages(library(stringr))

  # Check if the file is a parsed HTML
  if (typeof(html) != "externalptr"){
    stop(paste("Not a valid html file from author's citation website",
               "please check the format of the input file", sep = " "))
  }

   if (class(html)[[1]] != "HTMLInternalDocument") {
    stop(paste("Not a valid html file from author's citation website",
               "please check the format of the input file", sep = " "))
  }

  # Get the links inside the html
  links <- getHTMLLinks(html, baseURL = urlInit, relative = FALSE)

  # Check if there are valid links in the file
  if (length(links) <= 0){
    stop(paste("Not a valid html file from author's citation website",
               "the file does not contain any valid link", sep = " "))
  }

  # Generate the link containing the query
  url4queries <- paste("http://scholar.google.com", links[8], sep = '')

  # Global vectors for recording all the queries
  GTitles <- c()
  GJournals <- c()
  GAuthors <- c()
  GYears <- c()
  GCitations <- c()

  # Initial values: from (start) and size
  from <- 0
  size <- 100

  # Loop for getting all the articles based on the query structure
  while (TRUE){
    query <- paste("&cstart=",from,"&pagesize=",size,sep = '')
    urlq <- paste(url4queries,query,sep = '')
    html <- readLines(urlq)
    html <- htmlParse(html)
    # Articles titles
    Titles <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                          tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_t']
                          //a[@class = 'gsc_a_at']", xmlValue,
                          trim = TRUE, encoding = "UTF-8")

    # Break condition: no titles/papers inside the HTML object
    if (length(Titles) == 0 ) {
      break
    }

    # Journals and authors array
    JournalAuthors <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                                  tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_t']
                                  //div[@class = 'gs_gray']", xmlValue,
                                  trim = TRUE, encoding = "UTF-8")

    # Getting the Journals (even numbers of previos vector)
    Journals <- JournalAuthors[seq(2, length(JournalAuthors), 2)]

    # We delete the year of publication after the Journal
    Journals <- substring(Journals, 0, nchar(Journals)-6)

    # Getting the authors (odd numbers)
    Authors <- JournalAuthors[seq(1, length(JournalAuthors), 2)]

    # Years of publication
    Year <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//tr[@class ='gsc_a_tr']
                        //td[@class = 'gsc_a_y']//span[@class = 'gsc_a_h']",
                        xmlValue, trim = TRUE, encoding = "UTF-8")

    # Number of citations (No repetitions)
    Citations <- xpathSApply(html, "//tbody[@id = 'gsc_a_b']//
                             tr[@class ='gsc_a_tr']//td[@class = 'gsc_a_c']//a",
                             xmlValue, trim = TRUE, encoding = "UTF-8")

    # Delete potential non numerical elements by transforming and filtering the array
    Citations <- as.numeric(Citations)
    remove <- c(NA)

    # Re-Transform it to a character array for the DF
    Citations <- as.character(Citations[!Citations %in% remove])

    # Record in global vectors
    GTitles <- c(GTitles, Titles)
    GAuthors <- c(GAuthors, Authors)
    GJournals <- c(GJournals, Journals)
    GYears <- c(GYears, Year)
    GCitations <- c(GCitations, Citations)

    # Update the counter
    from <- from + 100
  }

  # Dimension checks: fill with 0 empty citations
```

```r
  if (length(GCitations) != length(GTitles)){
    GCitations <- c(GCitations, integer(length(GTitles) - length(GCitations)))
  }

  # Create a DataFrame
  x_name <- "Title"
  y_name <- "Authors"
  z_name <- "Journals"
  w_name <- "Year"
  v_name <- "Citations"

  require(reshape2)
  suppressMessages(ScholarDF <- melt(data.frame(GTitles, GAuthors, GJournals,
                                                GYears, GCitations)))
  colnames(ScholarDF) <- c(x_name, y_name, z_name, w_name, v_name)

  # Return the DataFrame
  return(ScholarDF)
}
```

Run the code

```r
# Call the function and create a DataFrame from the author's website
ResultsList <- CitesScholar("Geoffrey Hinton")

# Separate the UserID and Authorhtml (for simplicity)
UserID <- ResultsList[[1]]
Authorhtml <- ResultsList[[2]]

# Create the full data frame
FullScholarDF <- CreateDataFrame(Authorhtml)

# Display the length of the DataFrame (checking the length of any column)
length(FullScholarDF$Title)

# Printing a summary using the apply function (for a 'pretty' output)
apply(FullScholarDF, 2, function(x)
  if (length(x) > 15 && is.na(str_match(x, "^[[:digit:]]"))) {
    paste(substr(x, start = 0, stop = 15), "...", sep = "")
  }
  else {substr(x, start = 0, stop = length(x))
  }
)
```