1. First step is to load the respective documents (DEF 14A FILINGS) to be extracted and processed in a folder for that corresponding company on a yearly basis.

For instance, we create  a folder called datasets/txt.

**cname <- file.path (".", "datasets", "txt")**

We are ready to load files from the directory by using **Dirsource ()**

Following code shows the process: - library(tm)

**docs <- Corpus (DirSource(cname))**

Reading corpus pdf documents by the following code:-

**docs <- Corpus(DirSource(cname), readerControl=list(reader=readPDF))**

 Now that loading the data is done, Next would be to read the datasets.

1. We need to use the **readLines()** function to read the datasets, or alternatively  to read the HTML tags, we can use XML package library.

For Instance : **readLines ( “ex.data”,-n).**

(In the place of ex.data, we should give the url of the link from which we would want the data to be read.)

n :  The (maximal) number of lines to read. Negative values indicate that one should read up to the end of input on the connection.

1. Once the datasets is read, we need to scan through the dataset and get the required/correct data using a function called **grep()** command. Grep functions search for matches to argument pattern within each element of a character. With this function, we can find the  required table ( Execution Compensation table)

For Instance, the general syntax is

**grep [-options] patterns [filename].**

1. Once the required table is found, we need to use the **readHTMLTable()** function to read the tables. Once the table is read, we need to remove the noises, to do so we need to clean the data which is done using TM package.
2. The tm package supports text, pdf, Microsoft word and XML formats, we will either use text formatting or Even the Xml type formatting for this purpose.

Let us assume that we save the files in pdf format and then want to bring them to use in Text format,

We use the following code:

**System ("for f in \*.pdf; do pdftotext -enc ASCII7 -nopgbrk $f; done")**

We can list the sources by function: - **getsources().** In addition we can read document in say pdf format by the function: - **getreaders()** ##”readPDF”

We can (and should) inspect the documents using **inspect().** This will assure us that data has been loaded properly and as we expect.

1. This step allows us to make sure that there is no discrepancy in our data i.e. Numbers, capitalization, common words, punctuation, cases, removing stop words as they lack any analytic value which is otherwise preparing our texts for high quality text analysis. Normally **tm\_map ()** is the function used to perform this step.

We use the following code:-

**getTransformations()**

**## [1] "removeNumbers" "removePunctuation" "removeWords"**

**## [4] "stemDocument" "stripWhitespace"**

1. A document term matrix is simply a matrix with documents as the rows and terms as the columns and a count of the frequency of words as the cells of the matrix. We use DocumentTermMatrix()

To create the matrix:

**dtm <- DocumentTermMatrix(docs)**

**Convert the dtm matrix into a simple matrix:**

**m <- as.matrix (dtm)**

**Once converted into a standard matrix the usual write.csv () can be used to write the data to file.**

**write.csv (m, file="dtm.csv")**

**8.**      Finally after exporting this data into excel**.**We check for consistency of this data if a particular record is present for two different documents for two consecutive years. We consolidate the data by the same company for its filing as the same CEO. However we make sure that we add the consolidated data for unique rows only. Finally we summarize this data by aggregations or charts in R to get the desired output.

1. Copy the url of DEF 14-A form of the company who's data needs to be analyzed

2. Using R, store the url into a data vector

   url<- "http://www.sec.gov/Archives/edgar/data/1069202/000119312516527385/d107529ddefa14a.htm"

3. Use htmltab() function to assemble data frame from HTML table data.In our case we will be reading

  "Compensation Summary" table to extract CEO's name, salary, vested and option stock awards. To use htmltab() function, installing package "XML" package.

   read.html <- htmltab(doc = url, which = "//th[text() = 'Compensation Summary']")

4. Now we will be using gather() function to collect the data from the "Compensation Summary" table.To use this function

   we need to install "tidyr" and migrittr" package.

   read.html %<>% gather(key, value, -"Compensation Summary")

5. Extracting the required columns using the following,

   my.dataset <-read.html%>% separate(key, into = c("Year", "Salary", "Stock Vested Award","Stock Option Award"), sep = " >> ") %>% head

   we have the reuired data from the html table now exporting it to excel

6. Writing into Excel.For this we need to download "Xlsx" package.

write.xlsx(my.dataset, "c:/mydata.xlsx")

7. To create salary trend and stock ownership analysis use plot function and lines function

The lines() adds information to a graph.

The plot( ) plots the (x,y) points

 # h is keyword for histogram

The below code is an example of creating histogram chart

for(i in 1:length(c("h"))){

  heading = paste("type=",c("h")[i])

  plot(x, y, type="n", main=heading)

  lines(x, y, type=c("h")[i])

}

The motive of this exercise is to analyse the **Summary Compensation Table** of the DEF 14A filing, we need to automate the process of collecting data like CEO name, Salary, Vested Stock Awards, and Option Awards for all available years in the filing using the Summary Compensation table in the html file. Once we have collected the data we need to perform some analysis and calculations on it. This can be achieved using the following steps.

**Step 1**: Store the URL

url <-“<http://www.sec.gov/Archives/edgar/data/1288776/000130817914000114/lgoogle2014_def14a.htm>”

**Step 2**: Parse the HTML content using the following code

html.doc <- htmlParse(url)

**Step 3**: Read all the tables present in the HTML parsed document

html.tables <- readHTMLTable(html.doc)

**Step 4**: Create a String vector containing the sub string of the column names present in the Summary Compensation Table

checkVextor <- c("principal","position","Year","Salary","Bonus","Stock","awards","open","compensation","Total","$")

**Step 5**: The following code iterates through all the table’s column names present in the html.tables table vector and checks for the highest match with thecheckvector mentioned above and provides the summary compensation table’s index in the html.tables vector.

for (i in 1:length(html.tables)) {

  matchs <- 0

  colNameVector <-colnames(html.tables[[i]])

  for (j in 1:length(colNameVector)) {

    for(n in 1:length(checkVextor)){

      if(!is.null(colnames(html.tables[[i]][j])) && grepl(checkVextor[n], colnames(html.tables[[i]][j])))

        matchs <- matchs+1

    }

  }

  tempProbability <- matchs/length(checkVextor)

  if(tempProbability>=highestProbability)

   {

    highestProbability=tempProbability

    requiredTable <- i

  }

}

**Step 6**: Now as we have the required summary compensation table being narrowed down automatically, we now export the dataframe to an excel file to clean the data and import it back using the following code.(We do this in excel because cleaning a single table is simpler in excel though it is a manual task)

write.table(html.tables[[requiredTable]],file="summaryCompensation.csv")

google2014 <-read.csv(file="summaryCompensation.csv")

**Step 7**: Once we have the cleaned up and structured dataframe we can utilise various inbuilt functions in R for calculations, manipulations and analysis.