Utility tool for fetching PUBCHEM ID and SMILE string form the names of molecules from publicly available molecular database

Abstract:

The importance of making large data set of molecules and to retrieve their smiles strings and molecule ids. The provided R code demonstrates a comprehensive method for obtaining PubChem IDs and canonical SMILES notations for a list of molecules. Utilizing the httr and jsonlite libraries, the code defines a function, get_pubchem_info, that processes each molecule name by making API requests to the PubChem database. This function first retrieves the PubChem CID (Compound Identifier) and then uses this CID to fetch the canonical SMILES string, a standardized representation of the molecule's structure. The code handles errors gracefully, ensuring that API request failures do not disrupt the entire process. It reads molecule names from an input CSV file, processes each name to gather the necessary chemical information, and combines the results into a unified data frame. Finally, it writes the consolidated data to an output CSV file, facilitating easy analysis and further utilization. This R script is highly useful for researchers and chemists who need to automate the retrieval of molecular identifiers and structures from PubChem. It streamlines the data acquisition process, reduces manual lookup efforts, and ensures data consistency, thereby enhancing productivity and accuracy in chemical informatics studies. For the input file preparation, molecule names should be collected from any public domain database e.g. KNApSAcK (http://www.knapsackfamily.com/KNApSAcK/), IMPPAT (https://cb.imsc.res.in/imppat/basicsearch/phytochemical) etc. and save the file named as molecules.csv. The csv file must contain the molecule names header in the molecule column name as showed in figure 1.

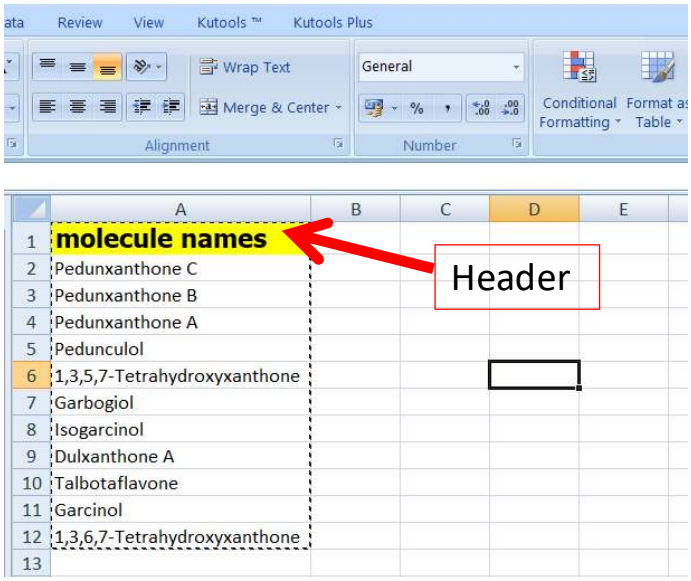
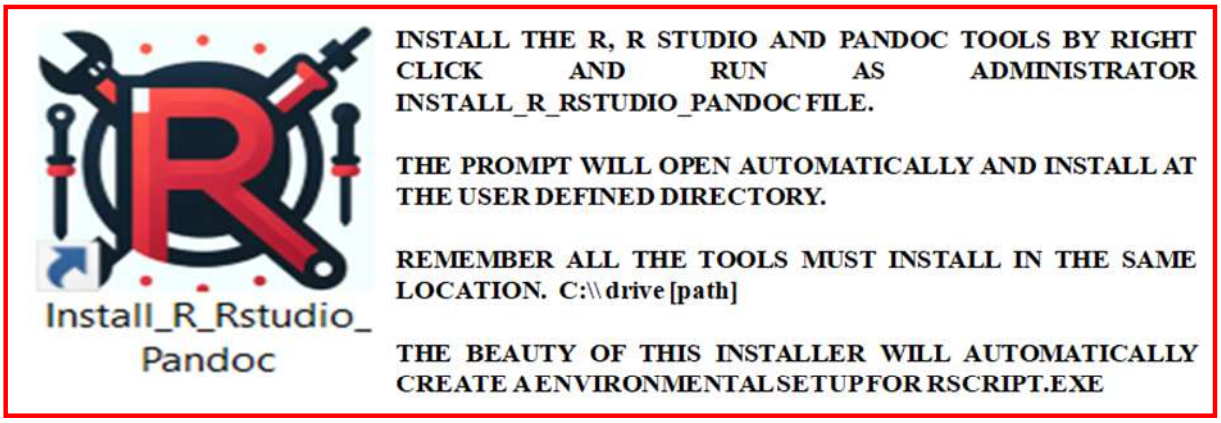


Figure 1. Typesetting of input csv file

Basic Installation of the tool

The zip file of fetching tool has to be unzipped first can be kept in any directory

STEP 1:



STEP 2:

STEP 3:

After this step please wait for 5-10 minutes, depending on the computer processor speed

Source Code:

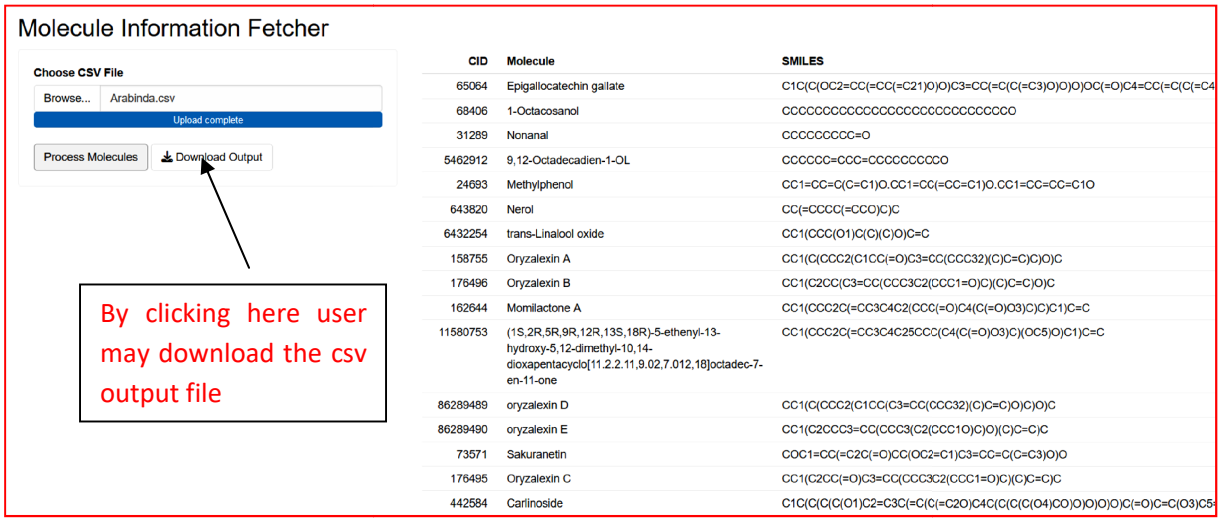
```
source("install_packages.R")

# Now load all the packages
invisible(lapply(required_packages, library, character.only = TRUE))

# Add this line at the beginning of your server.R script
cat("Current working directory is:", getwd(), "\n")

# Load necessary libraries
library(shiny)
library(httr)
library(jsonlite)

# Define UI for the application
ui <- fluidPage(
  titlePanel("Molecule Information Fetcher"),

  sidebarLayout(
    sidebarPanel(
      fileInput("file1", "Choose CSV File",
              accept = c("text/csv",
                         "text/comma-separated-values,text/plain",
                         ".csv")),
      actionButton("process", "Process Molecules"),
      downloadButton("downloadData", "Download Output")
    ),

    mainPanel(
      tableOutput("contents")
    )
  )
)

# Define server logic required to fetch PubChem data
server <- function(input, output, session) {

  get_pubchem_info <- function(molecule_name) {
    # URL encode the molecule name
    encoded_name <- URLencode(molecule_name, reserved = TRUE)

    # Construct URL to fetch PubChem CID
    url_cid <- paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/",
encoded_name, "/cids/JSON")

    # Fetch PubChem CID
    response_cid <- tryCatch(GET(url_cid), error = function(e) NULL)
```

```
      if (!is.null(response_cid) && status_code(response_cid) == 200) {
         content_cid <- content(response_cid, "parsed")
         pubchem_cid <- content_cid$IdentifierList$CID[1]

         if (!is.null(pubchem_cid)) {
            # Construct URL to fetch canonical SMILES using CID
            url_smiles <- paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/", pubchem_cid, "/property/CanonicalSMILES/JSON")

            # Fetch SMILES
            response_smiles <- tryCatch(GET(url_smiles), error = function(e) NULL)

            if (!is.null(response_smiles) && status_code(response_smiles) == 200) {
               content_smiles <- content(response_smiles, "parsed")
               smiles <- content_smiles$PropertyTable$Properties[[1]]$CanonicalSMILES

               if (!is.null(smiles)) {
                  return(data.frame(CID = pubchem_cid, Molecule = molecule_name, SMILES = smiles, stringsAsFactors = FALSE))
               }
            }
         }
      }

      return(data.frame(CID = NA, Molecule = molecule_name, SMILES = NA, stringsAsFactors = FALSE))
   }

   # Reactive value to store results
   results <- reactiveVal(NULL)

   observeEvent(input$process, {
      req(input$file1)
      inFile <- input$file1

      molecule_data <- tryCatch(read.csv(inFile$datapath, stringsAsFactors = FALSE), error = function(e) NULL)

      if (is.null(molecule_data)) {
         showNotification("Error reading input file.", type = "error")
         return(NULL)
      }

      molecule_names <- molecule_data$molecule
```

```
      if (is.null(molecule_names) || length(molecule_names) == 0) {
        showNotification("No molecule names found in the input file.", type = "error")
        return(NULL)
      }

      results_list <- lapply(molecule_names, get_pubchem_info)

      # Ensure column names are consistent for all data frames
      consistent_names <- c("CID", "Molecule", "SMILES")
      results_list <- lapply(results_list, function(df) {
        colnames(df) <- consistent_names
        return(df)
      })

      # Combine results into a single data frame
      results_df <- do.call(rbind, results_list)

      results(results_df)
    })

    output$contents <- renderTable({
      req(results())
      results()
    })

    output$downloadData <- downloadHandler(
      filename = function() {
        "molecule-output.csv"
      },
      content = function(file) {
        write.csv(results(), file, row.names = FALSE)
      }
    )
  }

# Run the application
shinyApp(ui = ui, server = server)
```

Output:

The code will generate an output file with Pubchem ID and Smiles as showed below



CID	Molecule	SMILES
65064	Epigallocatechin gallate	C1C(C(OC2=CC(=CC(=C21)O)O)C3=CC(=C(C(=C3)O)O)O)OC(=O)C4=CC(=C(C(=C4
68406	1-Octacosanol	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCO
31289	Nonanal	CCCCCCCCC=O
5462912	9,12-Octadecadien-1-OL	CCCCCC=CCC=CCCCCCCCCO
24693	Methylphenol	CC1=CC=C(C=C1)O.CC1=CC(=CC=C1)O.CC1=CC=CC=C1O
643820	Nerol	CC(=CCCC(=CCO)C)C
6432254	trans-Linalool oxide	CC1(CCC(O1)C(C)(C)O)C=C
158755	Oryzalexin A	CC1(C(CCC2(C1CC(=O)C3=CC(CCC32)(C)C=C)C)O)C
176496	Oryzalexin B	CC1(C2CC(C3=CC(CCC3C2(CCC1=O)C)(C)C=C)O)C
162644	Momilactone A	CC1(CCC2C(=CC3C4C2(CCC(=O)C4(C(=O)O3)C)C)C1)C=C
11580753	(1S,2R,5R,9R,12R,13S,18R)-5-ethenyl-13-hydroxy-5,12-dimethyl-10,14-dioxapentacyclo[11.2.2.11,9.02,7.012,18]octadec-7-en-11-one	CC1(CCC2C(=CC3C4C25CCC(C4(C(=O)O3)C)(OC5)O)C1)C=C
86289489	oryzalexin D	CC1(C(CCC2(C1CC(C3=CC(CCC32)(C)C=C)O)C)O)C
86289490	oryzalexin E	CC1(C2CCC3=CC(CCC3(C2(CCC1O)C)O)(C)C=C)C
73571	Sakuranetin	COC1=CC(=C2C(=O)CC(OC2=C1)C3=CC=C(C=C3)O)O
176495	Oryzalexin C	CC1(C2CC(=O)C3=CC(CCC3C2(CCC1=O)C)(C)C=C)C
442584	Carlinoside	C1C(C(C(C(O1)C2=C3C(=C(C(=C2O)C4C(C(C(C(O4)CO)O)O)O)O)C(=O)C=C(O3)C5=

******** Enjoy the tool for basic research********