3.9: R Assignment

Standardization

S. Kim, J. Cuadros

November 2nd, 2019

Objectives

- Understand the difference between compounds and substances in PubChem's terminology.
- Learn how chemical structures are represented in a real world.
- Understand the disambiguity of name-structure associations.
- Learn how to draw chemical structures programmatically.

In this task, we will use some cheminformatics packages to ease some processes. In R, some options are rcdk , ChemmineR and ChemmineOB . In Python, a useful package is RDKit ; in R, we'll make use of it online version, the Beaker API of ChEMBL (https://chembl.gitbook.io/chembl-interface-documentation/web-services).

Structure Standardization

PubChem contains more than 200 millions chemical records submitted by hundreds of data contributors. These depositor-provided records are archived in a database called "Substance" and each record in this database is called a substance. The records in the Substance database are highly redundant, because different data contributors may submit information on the same chemical, independently of each other. Therefore, PubChem extracts unique chemical structures from the Substance database through a process called standardization (https://doi.org/10.1186/s13321-018-0293-8). These unique structures are stored in the Compound database and individaual records in this database is called "compounds". To learn more about the PubChem compounds and substances, please read this PubChem Blog post (https://go.usa.gov/xVXct (https://go.usa.gov/xVXct)). The code cells below demonstrates the effects of chemical structure standardization.

Step 1. Download a list of the SIDs associated with a given CID First, let's get a list of SIDs that are associated CID 1174 (uracil).

```
pugrest <- 'https://pubchem.ncbi.nlm.nih.gov/rest/pug'
pugin <- 'compound/cid/1174'
pugoper <- 'sids'
pugout <- 'txt'

url <- paste(pugrest,pugin,pugoper,pugout,sep="/")
sids <- readLines(url)
length(sids)
```

The above request returns 300+ substances, all of which are standardized to the same structure (CID 1174).

Step 2. Download the structure data for the SIDs Now retrieve the depositor-provided structures for the returned substances.

```
chunk_size <- 50
num_chunks <- ceiling(length(sids)/chunk_size)

sdf <- character(0)
for(i in seq(num_chunks)) {

 print(paste("Processing chunk", i))

 idx1 <- chunk_size * (i - 1) + 1
```

```
 idx2 <- chunk_size * i
 str_sids <- paste(sids[idx1:min(idx2,length(sids))], collapse=",")

 url <- paste("https://pubchem.ncbi.nlm.nih.gov/rest/pug/substance/sid",
 str_sids, "record/sdf", sep="/")
 sdf <- c(sdf,readLines(url))
 Sys.sleep(0.2)
}
```

```
 writeLines(sdf,"uracil_from_sids.sdf")
```

Step 3. Convert the structures in the SDF file into the SMILES strings and identify unique SMILES and their frequencies.

```
if (!require("BiocManager", quietly=TRUE)) {
 install.packages("BiocManager", repos="https://cloud.r-project.org/",
 quiet=TRUE, type="binary")
library("BiocManager")
}
if (!require("ChemmineR", quietly=TRUE)) {
 BiocManager::install("ChemmineR",ask=FALSE)
 library("ChemmineR")
}
if (!require("ChemmineOB", quietly=TRUE)) {
 BiocManager::install("ChemmineOB",ask=FALSE)
 library("ChemmineOB")
}
```

```
 convertFormatFile("SDF","CAN","uracil_from_sids.sdf","uracil_from_sids.smi")

#convertFormatFile("SDF","CAN","https://chem.libretexts.org/@api/deki/files/346075/ura

smis <- read.table("uracil_from_sids.smi",header=F,sep="\t")
tabSMIs <- table(smis[,1])
tabSMIs <- sort(tabSMIs,decreasing = TRUE)
tabSMIs
```

The above output shows that the 300+ SIDs associated with CID 1174 are represented with six different SMILES strings. In addition, some substance records that resulted in an “empty” SMILES strings, implying that the depositors of these substance records did not provide structral information. You may want to what these substances are, but this can be recovered from the SDF file as follows.

```
uracilSDFset <- read.SDFset("uracil_from_sids.sdf")
```

```
 nonvalidSDF <- uracilSDFset[!validSDF(uracilSDFset)]
 unlist(lapply(nonvalidSDF@SDF,
```

```
      function(x) paste(x@datablock["PUBCHEM_SUBSTANCE_ID"],
      x@datablock["PUBCHEM_SUBS_AUTO_STRUCTURE"], sep=": ")))
```

Sometimes a data depositor does not provide the structure of a chemical but its chemical synonym(s). In that case, PubChem uses the chemical synonyms to assign a structure to this structure-less record. For example, SID 50608295 (one of the 12 structures without SMILES strings in the above output) did not have a depositorprovided structure, but its depositor-provided synonyms include “CID1174”. Therefore, PubChem assigns SID 50608295 to CID 1174, although the depositor did not provide the structure of SID 50608295. (Please check the structure and synonyms for SID 50608295 stored in the SDF file (“cid2sids-uracil.sdf”) generated in step 2).

Step 4. Generate the structure images from the SMILES

Now we want to see what these SMILES strings look like, by drawing molecular structures from them.

```
if(!require("httr", quietly=TRUE)) {
 install.packages("httr", repos="https://cloud.r-project.org/",
 quiet=TRUE, type="binary")
 library("httr")
}
if(!require("png")) {
 install.packages(("png"), repos="https://cloud.r-project.org/",
 quiet=TRUE, type="binary")
 library("png")
}
if(!require("grid")) {
 install.packages(("grid"), repos="https://cloud.r-project.org/",
 quiet=TRUE, type="binary")
 library("grid")
}
if(!require("gridExtra")) {
 install.packages(("gridExtra"), repos="https://cloud.r-project.org/",
 quiet=TRUE, type="binary")
 library("gridExtra")
}
```

```
 tabSMIs <- tabSMIs[names(tabSMIs)!=""]
names(tabSMIs)
```

```
 vecSMI <- names(tabSMIs)
names(vecSMI) <- names(tabSMIs)

sdf <- smiles2sdf(vecSMI)

# ChemmineR has a function to print the molecules, but
# results are far from acceptable
# png("plots.png",width=400,height=2000)
# ChemmineR::plot(sdf, griddim=c(ceiling(length((vecSMI))/2),2),
# regenCoords=TRUE,atomcex=1 )
```

```
#
# dev.off()
# A better option is to use the ChEMBL Beaker API (RDKit online)

write.SDF(sdf,"uracil_unique.sdf")
url <- "https://www.ebi.ac.uk/chembl/api/utils/ctab2image?size=300"
mydata <- list(sdf=upload_file("uracil_unique.sdf"))
res <- POST(url, body = mydata)
img <- readPNG(res$content, native=TRUE)
grid.arrange(rasterGrob(img))
writePNG(img,"uracil_unique.png")
```

You may want to write these molecule images in separate files.

```
dir.create("uracil_unique",showWarnings = FALSE)
fileSDF <- write.SDFsplit(sdf,"uracil_unique/",1)$filename
url <- "https://www.ebi.ac.uk/chembl/api/utils/ctab2image?size=300"

for(i in seq(fileSDF)) {
 mydata <- list(sdf=upload_file(fileSDF[i]))
 res <- POST(url, body = mydata)
 img <- readPNG(res$content, native=TRUE)
 writePNG(img,paste("uracil_unique/",
 formatC(i,format="d",width=2,flag="0"),".png",sep=""))
 Sys.sleep(1)
}

dir("uracil_unique",pattern=".*[.]png")
```

As shown these chemical images, the 300+ substances associated with CID 1174 (uracil) correspond to 6 tautomeric forms of uracil, which differ from each other in the position of “movable” hydrogen atoms. Compare these structures with their standardized structure (CID 1174).

```
img <- readPNG(GET('https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/1174/PNG?i
grid.arrange(rasterGrob(img))
```

Exercise 1a: The function used in Step 3 generates the canonical SMILES string by default. OpenBabel supports a second format (“SMI”) that does not use a canonical numbering when creating the SMILES. Other options can be added. Write a code that generates non-canonical SMILES strings for the 300+ substance records associated with uracil (CID 1174).

- Ignore/skip structure-less records using a conditional statement (i.e., an if statement).
- Print the number of unique non-canonical SMILES.
- Print unique non-canonical SMILES, sorted by frequency.

For a given molecule, there may be multiple ways to write SMILES strings: one of them is selected as the “canonical” SMILES and all the others are considered as “non-canonical”. However, for the purpose of this exercise, we want to generate only one non-canonical SMILES for each record (because the function will return only one SMILES string, the canonical SMILES or one of possible non-canonical SMILES).

```
#Write your code here.
```

Exercise 1b: NOT INCLUDED IN THE R VERSION

Exercise 1c: Retrieve the substance records associated with guanine (CID 135398634) and display unique structures generated from them, by following these steps:

- Retrieve the SIDs associated CID 135398634
- Download the structure data for the retrieved SIDs (in SDF)
- Generate canonical SMILES strings from the structure data in the SDF file and identify unique canonical SMILES strings
- Draw the structures represented by the unqiue canonical SMILES strings in a single figure.

```
#Write your code here.
```

Exercise 1d: Retrieve the substance records whose synonym is “glucose” and display unique structures generated from them, by following these steps: Retrieve the SIDs whose synonym is “glucose”. Download the structure data for the retrieved SIDs (in SDF) Generate canonical SMILES strings from the structure data in the SDF file and identify unique canonical SMILES strings Draw the structures represented by the unique canonical SMILES strings in a single figure.

```
#Write your code here.
```

Exercise 1e: Retrieve the compound records associated with the SIDs retrieved in Exercise 1d and display unique structures generated from them, by following these steps: Retrieve the CIDs associated with the SIDs whose name is “glucose”, using a single PUG-REST request

(i.e., using the list conversion covered in the previous activity, “Interconversion between PubChem records”).

- Identify unique CIDs from the returned CIDs.
- Retrieve the isomeric SMILES for the unique CIDs through PUG-REST.
- Draw the structures represented by the returned SMILES strings in a single figure.

```
#Write your code here.
```