Getting Molecular Properties through PUG-REST

S. Kim, J. Cuadros

August 4th, 2019

Objectives

- Learn the basic approach to getting data from PubChem through PUG-REST
- Retrieve a single property of a single compound.
- Retrieve a single property of multiple compounds
- Retrieve multiple properties of multiple compounds.
- Write a `for` loop to make the same kind of requests.
- Process a large amount of data by splitting them into smaller chunks

1. The Shortest Code to Get PubChem Data

Let's suppose that we want to get the molecular formula of water from PubChem through PUG-REST. You can get this data from your web browsers (Chrome, Safari, Internet Explorer, etc) via the following URL: https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/water/property/MolecularFormula/txt (https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/water/property/MolecularFormula/txt)
Getting the same data using a computer program is not very difficult. In R, this task can be with a single line of code.

```
readLines('https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/water/property/MolecularFo
rmula/txt',warn=FALSE)
```

```
## [1] "H2O"
```

The `readLines` function can read a text from a URL (or a file). The PUG-REST request URL, enclosed within a pair of single or double quotes(‘’ or “”), is provided within the parenteses. The second argument, `warn=FALSE` avoids a warning that appears when the data stream doesn't end with an empty line.

There are many other ways to get information from the web into R, for example using the function `GET` from the `httr` package, but for the moment we will stick to the simplest option.

As another example, the following code retrieves the number of heavy (non-hydrogen) atoms of butadiene.

```
readLines('https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/butadiene/property/HeavyAt
omCount/txt',warn=TRUE)
```

```
## [1] "4"
```

Exercise 1a: Retrieve the molecular weight of ethanol in a “text” format.

```
# Write your code here
```

Exercise 1b: Retrieve the number of hydrogen-bond donors of aspirin in a “text” format.

```
# Write your code here
```

2. Formulating PUG-REST request URLs using variables

In the previous examples, the PUG-REST request URLs were directly provided to the `readLines`, by explicitly typing the URL within the parentheses. However, it is also possible to provide the URL using a variable. The following example shows how to formulate the PUG-REST request URL using variables and pass it to `readLines`.

```
pugrest <- 'https://pubchem.ncbi.nlm.nih.gov/rest/pug'
pugin <- 'compound/name/water'
pugoper <- 'property/MolecularFormula'
pugout <- 'txt'

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

```
## [1] "https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/water/property/MolecularFormu
la/txt"
```

A PUG-REST request URL encodes three pieces of information (input, operation, output), preceded by the prologue commont to all requests. In the above code, these pieces of information are stored in four different variables (`pugrest`, `pugin`, `pugoper`, `pugout`) and combined into a new variable `url`.

Here, the strings stored in the four variables are joined by the “/” character as a separator.

Then, the url can be passed to `readLines`.

```
readLines(url)
```

```
## [1] "H2O"
```

Warning: Avoid using reserved or function names as a variable names. `in`, `c`, `t`, `names` ... are some examples of variable names to be avoided in `R`. In the example above, the variables are prefixed with “pug” to avoid naming conflicts.

3. Making multiple requests using a for loop

The approach in the previous section (that use variables to construct a request URL) looks very inconvenient, compared to the code shown at the beginning, where the request URL is directly provided to `readLines`. If you are making only one request, it would be simpler to provide the URL directly to `readLines`, rather than assiging the pieces to variables, constructing the URL from them, and passing it to the function. However, if you are making a large number of requests, it would be very time consuming to type the respective request URLs for all requests. In that case, you want to store common parts as variables and use them in a loop. For example, suppose that you want to retrieve the SMILES strings of 5 chemicals.

```
chemicals <- c('cytosine', 'benzene', 'motrin', 'aspirin', 'zolpidem')
```

Now the chemical names are stored in a character vector called `chemicals`. Using a `for` loop, you can loop over each chemical name, formulating the request URL and retrieving the desired data, as shown below.

```
pugrest <- "https://pubchem.ncbi.nlm.nih.gov/rest/pug"
pugoper <- "property/CanonicalSMILES"
pugout  <- "txt"

# loop over each element in the 'chemicals' list
for(chemical in chemicals) {
    pugin <- paste('compound/name/',chemical,sep="")
    url <- paste(pugrest,pugin,pugoper,pugout,sep="/")
    res <- readLines(url,warn=TRUE)
    print(paste(chemical, ": ", res,sep=""))
}
```

```
## [1] "cytosine: C1=C(NC(=O)N=C1)N"
## [1] "benzene: C1=CC=CC=C1"
## [1] "motrin: CC(C)CC1=CC=C(C=C1)C(C)C(=O)O"
## [1] "aspirin: CC(=O)OC1=CC=CC=C1C(=O)O"
## [1] "zolpidem: CC1=CC=C(C=C1)C2=C(N3C=C(C=CC3=N2)C)CC(=O)N(C)C"
```

Warning: When you make a lot of programmatic access requests using a loop, you should limit your request rate to or below **five requests per second**. Please read the following document to learn more about PubChem's usage policies: https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access$_RequestVolumeLimitations (https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access$_RequestVolumeLimitations)
Violation of usage policies may result in the user being **temporarily blocked** from accessing PubChem (or NCBI) resources**

In the for-loop example above, we have only five input chemical names to process, so it is not likely to violate the five-requests-per-second limit. However, if you have thousands of names to process, the above code will exceed the limit (considering that this kind of requests usually finish very quickly). Therefore, the request rate should be adjusted by using the **`Sys.sleep`** function. For simplcity, let's suppose that you have 12 chemical names to process (in reality, you could have much more to process).

```
chemicals <- c('water', 'benzene', 'methanol', 'ethene', 'ethanol',
               'propene','1-propanol', '2-propanol', 'butadiene',
               '1-butanol', '2-butanol', 'tert-butanol')
```

```
pugrest <- "https://pubchem.ncbi.nlm.nih.gov/rest/pug"
pugoper <- "property/CanonicalSMILES"
pugout  <- "txt"

# loop over each element in the 'chemicals' list
for(chemical in chemicals) {
    pugin <- paste('compound/name/',chemical,sep="")
    url <- paste(pugrest,pugin,pugoper,pugout,sep="/")
    res <- readLines(url,warn=TRUE)
    print(paste(chemical, ": ", res,sep=""))
    Sys.sleep(.3)
}
```

```
## [1] "water: O"
## [1] "benzene: C1=CC=CC=C1"
## [1] "methanol: CO"
## [1] "ethene: C=C"
## [1] "ethanol: CCO"
## [1] "propene: CC=C"
## [1] "1-propanol: CCCO"
## [1] "2-propanol: CC(C)O"
## [1] "butadiene: C=CC=C"
## [1] "1-butanol: CCCCO"
## [1] "2-butanol: CCC(C)O"
## [1] "tert-butanol: CC(C)(C)O"
```

It should be noted that the request volumn limit can be lowered through the dynamic traffic control at times of excessive load (https://pubchemdocs.ncbi.nlm.nih.gov/dynamic-request-throttling (https://pubchemdocs.ncbi.nlm.nih.gov/dynamic-request-throttling)). Throttling information is provided in the HTTP header response, indicating the system-load state and the per-user limits. Based on this throttling information, the user should moderate the speed at which requests are sent to PubChem. We will cover this topic later in this course.

Exercise 3a: Retrieve the XlogP values of linear alkanes with 1 ~ 12 carbons.
- Use the chemical names as inputs - Use a `for` loop to retrieve the XlogP value for each alkane. - Use the `Sys.sleep` function to stop the program for 10 seconds for every request.

```
# Write your code here
```

Exercise 3b: Retrieve the **isomeric** SMILES of the 20 common aminoacids. - Use the chemical names as inputs. Because the 20 common aminoacids in living organisms predominantly exist as one chrial form (the L-form), the names should be prefixed with **'L-'** (e.g., 'L-alanine', rather than 'alanine'), except for 'glycine' (which does not have a chiral center). - Use a `for` loop to retrieve the isomeric SMILES for each alkane. - Use the `Sys.sleep` function to stop the program for 10 seconds for every request.

```
# Write your code here
```

4. Getting multiple molecular properties

All the examples we have seen in this notebook retrieved a single molecular property for a single compound (although we were able to get a desired property for a group of compounds using a for loop). However, it is possible to get multiple properties for multiple compounds with a single request.

The following example retrieves the hydrogen-bond donor count, hydrogen-bond acceptor count, XLogP, TPSA for 5 compounds (represented by PubChem Compound IDs (CIDs) in a comma-separated values (CSV) format.

```
pugrest <- "https://pubchem.ncbi.nlm.nih.gov/rest/pug"
pugin   <- "compound/cid/4485,4499,5026,5734,8082"
pugoper <- "property/HBondDonorCount,HBondDonorCount,XLogP,TPSA"
pugout  <- "csv"

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

```
## [1] "https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/4485,4499,5026,5734,8082/prope
rty/HBondDonorCount,HBondDonorCount,XLogP,TPSA/csv"
```

```
# Print "-" 30 times (to print a line for readability)
cat(strrep("-",30))
```

```
## ------------------------------
```

```
read.table(url,sep=",",header=TRUE)
```

```
##    CID HBondDonorCount HBondDonorCount.1 XLogP  TPSA
## 1 4485               1                 1   2.2 110.0
## 2 4499               1                 1   3.3 110.0
## 3 5026               1                 1   4.3 123.0
## 4 5734               1                 1   0.2  94.6
## 5 8082               1                 1   0.8  12.0
```

In `R`, the `read.table` function allows reading formatted-text data files (or streams), such as CSV files. It returns a `data.frame`, which is a convenient data structure for data tables.

PubChem has a standard time limit of **30 seconds per request**. When you try to retrieve too many properties for too many compounds with a single request, it can take longer than the 30-second limit and a time-out error will be returned. Therefore, you may need to split the compound list into smaller chunks and process one chunk at a time.

```
cids <- c(443422, 72301, 8082, 4485, 5353740, 5282230, 5282138, 1547484,
          941361, 5734, 5494, 5422, 5417, 5290, 5245, 5026, 4746, 4507,
          4499, 4497, 4494, 4474, 4418, 4386, 4009, 4008, 3949, 3926, 3878,
          3784, 3698, 3547, 3546, 3336, 3333, 3236, 3076, 2585, 2520, 2351,
          2312, 2162, 1236, 1234, 292331, 275182, 235244, 108144, 104972, 77157,
          5942250, 5311217, 4564402, 4715169, 5311501)
chunk_size <- 10
num_chunks <- ceiling(length(cids) / chunk_size)

paste("# Number of CIDs:", length(cids) )
```

```
## [1] "# Number of CIDs: 55"
```

```
paste("# Number of chunks:", num_chunks )
```

```
## [1] "# Number of chunks: 6"
```

```
pugrest <- "https://pubchem.ncbi.nlm.nih.gov/rest/pug"
pugoper <- "property/HBondDonorCount,HBondDonorCount,XLogP,TPSA"
pugout  <- "csv"

pugin <- paste("compound/cid/",
               paste(cids[1:10],collapse=","),sep="")

url <- paste(pugrest,pugin,pugoper,pugout,sep="/")
df <- read.table(url,sep=",",header=TRUE)


for (i in 2:num_chunks) {
    idx1 = chunk_size * (i - 1) + 1
    idx2 = chunk_size * i

    pugin <- paste("compound/cid/",
                   paste(cids[idx1:pmin(idx2,length(cids))],collapse=","),sep="")

    url <- paste(pugrest,pugin,pugoper,pugout,sep="/")
    df <- rbind(df,read.table(url,sep=",",header=TRUE))
    Sys.sleep(10)
}

df
```

```
##        CID HBondDonorCount HBondDonorCount.1 XLogP  TPSA
## 1   443422               0                 0   3.1  40.2
## 2    72301               0                 0   3.2  40.2
## 3     8082               1                 1   0.8  12.0
## 4     4485               1                 1   2.2 110.0
## 5  5353740               2                 2   3.5  76.0
## 6  5282230               2                 2   3.2  84.9
## 7  5282138               1                 1   4.4 120.0
## 8  1547484               0                 0   5.8   6.5
## 9   941361               0                 0   6.0   6.5
## 10    5734               1                 1   0.2  94.6
## 11    5494               0                 0   5.0  57.2
## 12    5422               0                 0   6.4  61.9
## 13    5417               0                 0   3.2  40.2
## 14    5290               2                 2   2.6  62.2
## 15    5245               5                 5  -3.1 148.0
## 16    5026               1                 1   4.3 123.0
## 17    4746               1                 1   6.8  12.0
## 18    4507               1                 1   2.9 110.0
## 19    4499               1                 1   3.3 110.0
## 20    4497               1                 1   3.1 120.0
## 21    4494               1                 1   2.9 134.0
## 22    4474               1                 1   3.8 114.0
## 23    4418               1                 1   4.1  45.2
## 24    4386               2                 2   4.4  49.3
## 25    4009               2                 2   3.5  76.0
## 26    4008               1                 1   5.6 117.0
## 27    3949               0                 0   4.9  34.2
## 28    3926               1                 1   6.0  35.6
## 29    3878               2                 2   1.4  90.7
## 30    3784               1                 1   4.3 104.0
## 31    3698               2                 2  -0.2  68.0
## 32    3547               1                 1   1.0  70.7
## 33    3546               3                 3  -0.5 132.0
## 34    3336               1                 1   5.5  12.0
## 35    3333               1                 1   3.9  64.6
## 36    3236               0                 0   3.8  20.3
## 37    3076               0                 0   3.1  84.4
## 38    2585               3                 3   4.2  75.7
## 39    2520               0                 0   3.8  64.0
## 40    2351               0                 0   5.3  15.7
## 41    2312               0                 0   4.6  12.5
## 42    2162               2                 2   3.0  99.9
## 43    1236               1                 1   6.8 114.0
## 44    1234               0                 0   3.8  73.2
## 45  292331               2                 2   3.9  49.3
## 46  275182               1                 1   6.1  72.9
## 47  235244               1                 1   6.7  72.9
## 48  108144               2                 2   3.9 117.0
## 49  104972               1                 1   3.3  72.7
## 50   77157               1                 1   3.2  49.8
## 51 5942250               2                 2   3.5  76.0
## 52 5311217               1                 1   4.5  90.9
## 53 4564402               0                 0   4.1  45.5
## 54 4715169               2                 2  -1.6  63.3
## 55 5311501               0                 0   4.4  43.7
```

Exercise 4a: Below is the list of CIDs of known antiinflmatory agents (obtained from PubChem via the URL: https://www.ncbi.nlm.nih.gov/pccompound?LinkName=mesh_pccompound&from_uid=68000893 (https://www.ncbi.nlm.nih.gov/pccompound?LinkName=mesh_pccompound&from_uid=68000893)). Download the following properties of those compounds in a comma-separated format: Heavy atom count, rotatable bond count, molecular weight, XLogP, hydrogen bond donor count, hydrogen bond acceptor count, TPSA, and isomeric SMILES.

- Split the input CID list into small chunks (with a chunk size of 100 CIDs).
- Process one chunk at a time using a for loop.
- Do not forget to add `Sys.sleep` to comply the usage policy.

```
cids <-  c(471, 1981, 2005, 2097, 2151, 2198, 2206, 2214, 2244, 2307, 2308, 2313,
2355, 2396, 2449, 2462, 2466, 2581, 2662, 2794, 2863, 3000, 3003, 3033, 3056,
3059, 3111, 3177, 3194, 3230, 3242, 3282, 3308, 3332, 3335, 3342, 3360, 3371,
3379, 3382, 3384, 3394, 3495, 3553, 3612, 3672, 3715, 3716, 3718, 3778, 3824,
3825, 3826, 3935, 3946, 3965, 4009, 4037, 4038, 4044, 4075, 4159, 4237, 4386,
4409, 4413, 4487, 4488, 4495, 4534, 4553, 4614, 4641, 4671, 4692, 4781, 4888,
4895, 4921, 5059, 5090, 5147, 5161, 5208, 5228, 5339, 5352, 5359, 5362, 5468,
5469, 5475, 5480, 5509, 5733, 5743, 5744, 5745, 5753, 5754, 5755, 5834, 5865,
5875, 5876, 5877, 6094, 6213, 6215, 6247, 6436, 6741, 7090, 7497, 8522, 9053,
9231, 9642, 9782, 9878, 10114, 10154, 10170, 10185, 10206, 12555, 12938, 13802,
14982, 15209, 16490, 16533, 16623, 16639, 16752, 16923, 17198, 19161, 20469,
21102, 21700, 21800, 21826, 21975, 22419, 23205, 26098, 26248, 26318, 28718,
28871, 30869, 30870, 30951, 31307, 31378, 31508, 31635, 31799, 31800, 32153,
32327, 32798, 33958, 35375, 35455, 35935, 36833, 37425, 38081, 38503, 39212,
39941, 40000, 40632, 41643, 43261, 44219, 47462, 47795, 50294, 50295, 51717,
54445, 54585, 57782, 59757, 60164, 60490, 60542, 60712, 60726, 60864, 61486,
62074, 62924, 63006, 63019, 64704, 64738, 64746, 64747, 64927, 64945, 64971,
64982, 65394, 65464, 65655, 65679, 65762, 66249, 67417, 68700, 68704, 68706,
68731, 68749, 68819, 68865, 68869, 68917, 71246, 71354, 71364, 71386, 71398,
71414, 71415, 71771, 72158, 72300, 73400, 82153, 84003, 84429, 90763, 91626,
91670, 100472, 102011, 104762, 104943, 107641, 107738, 107793, 108068, 108130,
114753, 114840, 114917, 114999, 115239, 119032, 119286, 119365, 119607, 119828,
119871, 121928, 121957, 122139, 122179, 122182, 123619, 123673, 123723, 124978,
128191, 128229, 128571, 133021, 134896, 146364, 151075, 151166, 152165, 155354,
155761, 156391, 158103, 159557, 162666, 164676, 167928, 168928, 174093, 174277,
176155, 177976, 180604, 183088, 189821, 192156, 196122, 196840, 196841, 200674,
201587, 219121, 222786, 229860, 235244, 236702, 259846, 263373, 275182, 292331,
425990, 439503, 439533, 441335, 441336, 442534, 442993, 443943, 443949, 443967,
444036, 445154, 445858, 446925, 479503, 485711, 490428, 501254, 522325, 546807,
578771, 584547, 610479, 633091, 633097, 636374, 636398, 656604, 656656, 656852,
657238, 667550, 927704, 969510, 969516, 1548887, 1548910, 2737488, 3033890,
3033980, 3045402, 3051696, 3055172, 4129359, 4306515, 4483645, 5018304, 5185849,
5280802, 5280914, 5280915, 5281004, 5281071, 5281515, 5281522, 5281792, 5282183,
5282193, 5282230, 5282387, 5282402, 5282492, 5283542, 5283734, 5284538, 5284539,
5311051, 5311052, 5311066, 5311067, 5311093, 5311101, 5311108, 5311169, 5311180,
5318517, 5320420, 5322111, 5352624, 5353725, 5353726, 5353740, 5353864, 5354499,
5377381, 5420804, 5420805, 5458396, 5472495, 5481958, 5701991, 5702036, 5702148,
5702212, 5702252, 5702287, 5745214, 5942250, 6420050, 6429274, 6437368, 6437387,
6438873, 6447131, 6453785, 6473881, 6509979, 6708733, 6710677, 6714002, 6917783,
6917852, 6917894, 6918172, 6918173, 6918332, 6918445, 6918452, 6918612, 6925666,
7060958, 7251185, 9554199, 9798098, 9799453, 9841438, 9843941, 9846332, 9865808,
9868219, 9869053, 9871508, 9875547, 9883509, 9897518, 9897771, 9907157, 9913795,
9919776, 9926694, 9934547, 10363606, 10918539, 11158972, 11513733, 11561674,
11616712, 11870423, 11949636, 11954221, 11954316, 11954353, 11954369, 11957468,
11961431, 11972243, 11972532, 12300053, 12313906, 12313911, 12606303, 12634263,
12714644, 12874922, 13018150, 13020033, 13041095, 14010989, 14515707, 14798494,
15895902, 16051947, 16132369, 16213022, 16213698, 16218996, 16219353, 16220118,
16759566, 16760658, 17750985, 17753757, 18526330, 18632363, 18647121, 18943026,
20054915, 21120116, 21637635, 21637642, 21893738, 21893804, 21982135, 22141508,
22811280, 23509770, 23631982, 23653552, 23657872, 23663407, 23663409, 23663418,
23663959, 23663989, 23665411, 23665999, 23667642, 23669636, 23674183, 23674255,
23674745, 23675763, 23680530, 23681059, 23684814, 23688663, 23693301, 23694214,
23702389, 24181458, 24721429, 24761485, 24799587, 24847961, 24847981, 24867460,
24867465, 24867475, 24883465, 24916955, 25077872, 25113755, 25796773, 40469526,
44119558, 44202892, 44260118, 44266812, 44386560, 45006151, 45006158, 45039955,
45356876, 45356931, 45357558, 45357932, 45358013, 45358120, 45358130, 45358140,
45358148, 45358149, 45488525, 46174093, 46397498, 46780650, 46780910, 46783539,
```

```
46783786, 46783814, 46863906, 46878350, 46882877, 50989825, 51026956, 51340230,
51398089, 53384387, 53394893, 53486221, 53486290, 53486322, 54194814, 54605501,
54675840, 54676228, 54677470, 54677971, 54677972, 54677977, 54682045, 54684589,
54690031, 54697648, 54708862, 54714524, 56841932, 56842111, 56845155, 57347755,
57486087, 67668959, 67804972, 67986221, 70470286, 70678885, 71306882, 71587162,
72774967, 72941490, 72941625, 73758129, 73759663, 73759808, 74787565, 77906397,
78577433, 90488794, 91711382, 91826463, 91873711, 91881846, 92131836, 92462493,
102004404, 102601886, 117072385, 117072403, 117072410, 118701141, 118701402,
118984459, 122130078, 122130111, 122130185, 122130213, 122130768, 122173054,
122173183, 122361610, 123134657, 124081055, 124463365, 126968472, 126968501,
126968801, 126969212, 126969455, 129009998, 129010022, 129010033, 129010043,
129316829, 129317859, 129317898, 129628207, 129628892, 129670532, 129735029,
131632430, 131635023, 131676243, 131750284, 131954647, 131954667, 132399051,
132399058, 133112890, 133126366, 133126370, 133562807, 133659920, 133687604,
134129698, 134159361, 134460917, 134612785, 134687786, 134688123, 134688323,
134688977, 134689786, 134693106, 134693125, 134693234, 134694728, 134694860,
135413496, 135413505, 135414247, 135484078, 135515521, 135565709, 136040192,
137177332, 137699687, 137705034, 137705717, 137705725, 137705994, 137706376,
137706400, 137795135, 138059757, 138107776, 138113311, 138113507, 138113581,
138114182, 138114743)

length(cids)
```

```
## [1] 708
```

```
# Write your code here
```