ABSTRACT
The National Center for Biotechnology Information (NCBI) of the United States National Institute of Health has multiple databases that are generally accessed via a web interface. Popular databases include PubMed, which houses citations for biomedical literature, and PubChem, which houses chemical information, such as chemicals by name, molecular formula, or structure, among other data. NCBI provides a set of eight server-side programs, the Entrez Programming Utilities (E-utilities), that provide a stable interface into the Entrez query and database system. NCBI also made available the NCBI Datasets command line tools datasets and dataformat. These tools are helpful for work with sequences of SARS-CoV-2, for instance. The goal of this paper is to introduce the E-utilities and describe two SAS System® macros that use the HTTP procedure using the E-utilities URL and the command line tools.

INTRODUCTION
In observational research and clinical trials, one often uses databases to support the tasks a hand. For instance, in reading a Protocol, one may wish to obtain papers on the indication or treatment. One may need to convert a lab analyte from one unit to the another. Often, the web interfaces suffice, and search terms may not be complex. The need to alter or update the search based on the results of a process (program) or a large task necessitates a programmatic approach.

PubMed is a database of citations, abstracts, and, in some cases, full-text articles for biomedical literature that is freely available from the National Center for Biotechnology Information (NCBI) of the United States National Institute of Health (NIH). PubChem is another freely available NCBI database that contains information on chemicals such name, molecular formula, or structure, patents, among other data. The NCBI provides a set of eight server-side programs, the Entrez Programming Utilities (E-utilities), that provide a stable interface into the Entrez query and database system.

The advantages of using E-utilities included building searches based on the results of a program, which can include choosing useful search field tags. For instance, the author wrote a suite of programs (not presented) to that aligned the peptides in a protein of interest with peptides in other proteins in the reference genome, then searched various NCBI databases, for papers, SNPs (single nucleotides polymorphisms), and other information to investigate immunogenicity (ADA, anti-drug antibodies). At that time, the program ran for hours, if not days, so expanding it, for example, to the microbiome would become too cumbersome and extremely difficult to Validate to do manually. Another potential use would be to obtain a list of lab analytes that do not appear in one's internal databases and to search PubChem for a synonyms and molecular masses for conversion of lab data to controlled terminology, i.e. LBTEST, and preferred units, i.e. to convert mg/dL to mmol/L.

The E-Utility applications are:

- **EInfo** (database statistics)
  Provides the number of records indexed in each field of a given database, the date of the last update of the database, and the available links from the database to other Entrez databases.

- **ESearch** (text searches)
  Responds to a text query with the list of matching UIDs in a given database (for later use in ESummary, EFetch or ELink), along with the term translations of the query.

- **EPost** (UID uploads)
  Accepts a list of UIDs from a given database, stores the set on the History Server, and responds
with a query key and web environment for the uploaded dataset.

- **ESummary** (document summary downloads)
  Responds to a list of UIDs from a given database with the corresponding document summaries.

- **EFetch** (data record downloads)
  Responds to a list of UIDs in a given database with the corresponding data records in a specified format.

- **ELink** (Entrez links)
  Responds to a list of UIDs in a given database with either a list of related UIDs (and relevancy scores) in the same database or a list of linked UIDs in another Entrez database; checks for the existence of a specified link from a list of one or more UIDs; creates a hyperlink to the primary LinkOut provider for a specific UID and database, or lists LinkOut URLs and attributes for multiple UIDs.

- **EGQuery** (global query)
  Responds to a text query with the number of records matching the query in each Entrez database.

- **ESpell** (spelling suggestions)
  Retrieves spelling suggestions for a text query in a given database.

- **ECitMatch** (batch citation searching in PubMed)
  Retrieves PubMed IDs (PMIDs) corresponding to a set of input citation strings.

**Table 1** presents the databases on which these applications work with their Entrez database name and UID (unique identifier) common name as listed in the documentation. The Entrez name corresponds to the names found in the “All Databases” drop-down list to the left of the Search open-text bar on the main webpage (https://www.ncbi.nlm.nih.gov/), which can be used to explore and learn or confirm the results of using E-Utilities.

NCBI Datasets command line tools datasets and dataformat can be accessed within SAS using the PIPE engine to the FILENAME statement. These tools add immense versatility to SAS programs, especially those that require the large data of genomic investigations into SARS-CoV-2.

The ease of Validating a SAS System® program with its (dated) stored log and the ease of updating or replicating a search are important reasons to approach such tasks programmatically. The SAS System® provides many tools and appears to be fast enough to be practical for the tasks described. The goal of this paper is to introduce the E-utilities and describe a SAS macro that uses the HTTP procedure using the E-utilities URL and one that uses the NCBI command line tools datasets and dataformat.
Table 1 The E-utility Database Names and their respective Entrez Databases and UID common names.

<table>
<thead>
<tr>
<th>E-utility Database Name</th>
<th>Entrez Database</th>
<th>UID common name</th>
</tr>
</thead>
<tbody>
<tr>
<td>bioproject</td>
<td>BioProject</td>
<td>BioProject ID</td>
</tr>
<tr>
<td>biosample</td>
<td>BioSample</td>
<td>BioSample ID</td>
</tr>
<tr>
<td>biosystems</td>
<td>Biosystems</td>
<td>BSID</td>
</tr>
<tr>
<td>books</td>
<td>Books</td>
<td>Book ID</td>
</tr>
<tr>
<td>cdd</td>
<td>Conserved Domains</td>
<td>PSSM-ID</td>
</tr>
<tr>
<td>gap</td>
<td>dbGaP</td>
<td>dbGaP ID</td>
</tr>
<tr>
<td>dbvar</td>
<td>dbVar</td>
<td>dbVar ID</td>
</tr>
<tr>
<td>epigenomics</td>
<td>Epigenomics</td>
<td>Epigenomics ID</td>
</tr>
<tr>
<td>nucest</td>
<td>EST</td>
<td>GI number</td>
</tr>
<tr>
<td>gene</td>
<td>Gene</td>
<td>Gene ID</td>
</tr>
<tr>
<td>genome</td>
<td>Genome</td>
<td>Genome ID</td>
</tr>
<tr>
<td>gds</td>
<td>GEO Datasets</td>
<td>GDS ID</td>
</tr>
<tr>
<td>geoprofiles</td>
<td>GEO Profiles</td>
<td>GEO ID</td>
</tr>
<tr>
<td>nucgss</td>
<td>GSS</td>
<td>GI number</td>
</tr>
<tr>
<td>homologene</td>
<td>HomoloGene</td>
<td>HomoloGene ID</td>
</tr>
<tr>
<td>mesh</td>
<td>MeSH</td>
<td>MeSH ID</td>
</tr>
<tr>
<td>toolkit</td>
<td>NCBI C++ Toolkit</td>
<td>Toolkit ID</td>
</tr>
<tr>
<td>ncbisearch</td>
<td>NCBI Web Site</td>
<td>Web Site ID</td>
</tr>
<tr>
<td>nlmcatalog</td>
<td>NLM Catalog</td>
<td>NLM Catalog ID</td>
</tr>
<tr>
<td>nuccore</td>
<td>Nucleotide</td>
<td>GI number</td>
</tr>
<tr>
<td>omia</td>
<td>OMIA</td>
<td>OMIA ID</td>
</tr>
<tr>
<td>popset</td>
<td>PopSet</td>
<td>PopSet ID</td>
</tr>
<tr>
<td>probe</td>
<td>Probe</td>
<td>Probe ID</td>
</tr>
<tr>
<td>protein</td>
<td>Protein</td>
<td>GI number</td>
</tr>
<tr>
<td>proteinclusters</td>
<td>Protein Clusters</td>
<td>Protein Cluster ID</td>
</tr>
<tr>
<td>pcassay</td>
<td>PubChem BioAssay</td>
<td>AID</td>
</tr>
<tr>
<td>pccompound</td>
<td>PubChem Compound</td>
<td>CID</td>
</tr>
<tr>
<td>pcsubstance</td>
<td>PubChem Substance</td>
<td>SID</td>
</tr>
<tr>
<td>Pubmed</td>
<td>PubMed</td>
<td>PMID</td>
</tr>
<tr>
<td>Pmc</td>
<td>PubMed Central</td>
<td>PMCID</td>
</tr>
<tr>
<td>Snp</td>
<td>SNP</td>
<td>rs number</td>
</tr>
<tr>
<td>Sra</td>
<td>SRA</td>
<td>SRA ID</td>
</tr>
<tr>
<td>structure</td>
<td>Structure</td>
<td>MMDB-ID</td>
</tr>
<tr>
<td>taxonomy</td>
<td>Taxonomy</td>
<td>TaxID</td>
</tr>
<tr>
<td>Unigene</td>
<td>UniGene</td>
<td>UniGene Cluster ID</td>
</tr>
<tr>
<td>Unists</td>
<td>UniSTS</td>
<td>STS ID</td>
</tr>
</tbody>
</table>
METHODS
WEB INTERFACES TO NCBI DATABASES

Even if the user is experienced using PubMed, referencing the “User Guide” and “Entrez Help” is highly recommended. When accessing Pubmed through its web interface, one can use the “Advanced” button to build a search with tags. In general, this is not needed, but demonstrates the process, which is helpful for learning how to build the URL for E-Utilities. Display 1 provides an example of a PubMed search built with the “Advanced” button.

Display 1 Building a search using the “Advanced” button on the web interface of PubMed.

The search built in Display 1 returns (as of April 12, 2022) the single paper as shown in Display 2. The author would have manually typed this search as “Wen[au] cytoskeleton[ti] Covid-19” into the web interface, noting that the quotation marks only delineate the search term in this paper and that they have the purpose of making their contents and exact match in PubMed. Further note that “AND” is implied, and parentheses as most SAS programmers and mathematicians might expect. Consider: “(cytoskeleton[ti] OR microtubule [ti]) Covid-19”. The reader who is new to PubMed may wish to contrast the results of “host cell receptor” [ti] Covid-19 and “host cell” [ti] Covid-19 (without the single quotations marks when typed or copied to Pubmed.)
E-UTILITES (ENTREZ PROGRAMMING UTILITIES)

The reader is encouraged to read or at least peruse “Entrez Programming Utilities Help”\(^3\). To obtain a list of databases, one can submit the following URL in a web browser:


Alternatively, calling the M_U_EUTILS SAS macro in Appendix 1 with the following macro parameter and value invokes one of its two special uses (the other being HELP):

%m_u_eutils
   {"list_current_dbs = Y }

and generates the list of the available databases are in Display 3 in the log (as of April 12, 2022). Note that the current list differs from this list provided in the documentation. The macro does not parse the code with the XML engine to the LIBNAME statement, but rather abstracts the names with a “quick and dirty” regular expression using the SAS PRX functions.
NOTE: 50 records were read from the infile RESP.
The minimum record length was 0.
The maximum record length was 126.

NOTE: DATA statement used (Total process time):
  real time           0.14 seconds
  cpu time            0.01 seconds

Display 3 The databases returned by the E-Utilities program EINFO as of April 22, 2022.

EXAMPLES

Using the HELP macro parameter, one can find some usage rules. To examine the URL created and exit
the macro with no further action, one can use the macro parameters DEBUG_URL:

```bash
%m_u_eutils
  (e_utilities = einfo.fcgi,
   debug_url = Y)
```


To examine the XML results without providing code for the CODE macro parameter, one can invoke the
macro parameters DEBUG_RETURN:

```bash
%m_u_eutils
  (e_utilities = einfo.fcgi,
   debug_return = Y)
```

The following call obtains suggestions for spelling using the database pubmed:

```perl
%m_u_eutils ( e_utilities = espell.fcgi , db = pubmed , term = acetaminophin , api_key = , code = __rc1 = prxparse( "/<Query>(.+)</Query>" ) %str(); __rc2 = prxparse( "/<CorrectedQuery>(.+)</CorrectedQuery>" ) %str(); retain __rc %str();

file print %str();

length query correctedquery $ 200 %str();
if prxmatch( __rc1 , _infile_ )
    do %str();
        query = prxposn( __rc1 , 1 , _infile_ ) %str();
        put "Query= " @25 query %str();
    end %str();
if prxmatch( __rc2 , _infile_ )
    do %str();
        correctedquery = prxposn( __rc2 , 1 , _infile_ ) %str();
        put "CorrectedQuery= " @25 CorrectedQuery %str();
    end %str();
)

Query= acetaminophin
CorrectedQuery= acetaminophen
```

The call below replicates the search described in Display 2:

```perl
%m_u_eutils ( e_utilities = esearch.fcgi , db = pubmed , term = Wen[Author]+AND+cytoskeleton[Title]+AND+Covid-19 , api_key = , debug_return = Y , code = __rc = prxparse( "/<Id>([a-zA-Z0-9]+)</Id>/" ) %str();

retain __rc %str();

file print %str();

length pmid $ 20 %str();
if prxmatch( __rc , _infile_ )
    do %str();
        pmid = prxposn( __rc , 1 , _infile_ ) %str();
        put pmid %str();
    end %str();
)
```

7
As of April 12, 2022, this code returns the single PubMed ID “32717049”. Note that providing code in such a way has some issues, such as the substitution of semi-colons with their masked versions "%str(;)". The author removed his API Key, but had used his valid key.

To demonstrate the use the history server to replicate the search in the code above two calls to the macro were used. The second uses the QUERY_KEY and WEBENV values generated by the following call:

```plaintext
%symdel QueryKey_return
   webenv_return
;
%m_u_eutils
   "e_utilities" = esearch.fcgi
   db = pubmed
   term = Wen[Author]+AND+Covid-19
   api_key =
   usehistory = Y
   debug_return = Y
   code =
      __rc1 = prxparse( "/<QueryKey>(.+)" ) %str(;)
      __rc2 = prxparse( "/<WebEnv>(.+)" ) %str(;)
      retain __rc1 __rc2
      length querykey webenv $ 200 %str(;)
      if prxmatch( __rc1 , _infile_ ) then
         do %str(;)
            querykey = prxposn( __rc1 , 1 , _infile_ ) %str(;)
            call symputx( "querykey_return" , querykey ) %str(;)
         end %str(;)
      if prxmatch( __rc2 , _infile_ ) then
         do %str(;)
            webenv = prxposn( __rc2 , 1 , _infile_ ) %str(;)
            call symputx( "webenv_return" , webenv ) %str(;)
         end %str(;)
   ) ;
```

When the usehistory = y, the macro executes the %GLOBAL statement with for QUERYKEY_RETURN and WEBENV_RETURN. To demonstrate this, these global macro variables were deleted prior to call the macro. The code below runs a second search using these two macro variables and the result is the intersection of the two separate searches:
As of April 12, 2022, this code returns the single PubMed ID “32717049”, matching the first example and the search using the PubMed web interface.

A frequent task in clinical trials programming is to convert lab test from original to prefer units. Obtaining the list manually is cumbersome and may result in errors. Using the pccompound database might help, but the user may also need to familiarize her- or himself with this resource. The following code, consisting of two calls to the macro, demonstrates an attempt to find the molecular mass of acetaminophen:

```%m_u_eutils
  { e_utilities   = esearch.fcgi
    db           = pccompound
    term         = acetaminophen
    api_key      =
    usehistory   =
    debug_return = Y
    code         =
  };

%str();

if prxmatch( __rc , _infile_ )
  then
    do %str();
      querykey = prxposn( __rc
                         , _infile_ ) %str();
      call symputx( "querykey_return"
                    , querykey ) %str();
    end %str();

if prxmatch( __rc2 , _infile_ )
  then
    do %str();
      we bev = prxposn( __rc2
                        , _ infile_ ) %str();
    end %str();
```

```%m_u_eutils
  { e_utilities   = esearch.fcgi
    db           = pccompound
    term         = acetaminophen
    api_key      =
    usehistory   =
    debug_return = Y
    code         =
  };

%str();

if prxmatch( __rc , _infile_ )
  then
    do %str();
      querykey = prxposn( __rc
                         , _infile_ ) %str();
      call symputx( "querykey_return"
                    , querykey ) %str();
    end %str();

if prxmatch( __rc2 , _infile_ )
  then
    do %str();
      we bev = prxposn( __rc2
                        , _ infile_ ) %str();
```
The record returns includes the following, which the user may parse or attempt to refine using options:

```
<item name="IUPACName" type="string">N-(4-hydroxyphenyl)acetamide;methanol</item>
<item name="string" type="string">acetaminophen:methanol solvate</item>
<item name="string" type="string">paracetamol:methanol solvate, (1:1)</item>
<item name="MolecularWeight" type="string">549.600</item>
```

NCBI COMMAND LINE TOOLS

Certain information for the datasets and dataformat tools is available from the tools themselves, but the interested reader should at least peruse the reference. The following code demonstrates how to obtain help from the tool using the macro presented in Appendix 2:

```
%m_ncbi_cli_tool
( cli_tool = dataformat
, cli_tool_path = ~\dataformat.exe
, options = help tsv genome
)
```

The user will have to provide the path. Note that one can build the options incrementally, for instance, the author learned of the "genome" Report Command to the tsv command by running the code above without "genome" then adding it.

To code below mimics the example:

```
%m_ncbi_cli_tool
( options = %str(download gene gene-id 1,2,3,9,10,11,12,13,14,15,16,17 --filename cli_tools_test.zip)
, api_key =
, pre = cd /D C:\Users\kviel\OneDrive\Documents\12_NIH\NCBI\NLM\nuccore\SARS_COV_2\datasets %nrstr(&&)
)
```

Note that the PRE macro parameter changed the directory using the MS-DOS command cd before the executable on the command line. The ampersand is the line continuation in the MS-DOS command line but it needed to be masked. The zip file was 143 KB and unzipped to 474 KB and seven files.

The format of the reports is jsonl, but the dataformat tool provides a way to convert it. For example, the following code:

```
%m_ncbi_cli_tool
( cli_tool = dataformat
, cli_tool_path = ~\dataformat.exe
, pre = cd /D C:\Users\kviel\OneDrive\Documents\12_NIH\NCBI\NLM\nuccore\SARS_COV_2\datasets\cli_tools_test %nrstr(&&)
)
```

results in following snippet being written to the log:
The NCBI has made an incredible resource freely available which is indispensable to researchers investigating the SARS-CoV-2 pandemic. The following code will download every DNA (RNA) sequence in the database:

```bash
%ncbi_cli_tool
  ( options = download virus genome taxon sars-cov-2 --exclude-cds --exclude-protein --exclude-gff --exclude-pdb --filename sar2_genome_20220316.zip
    , api_key =
    , psn = )
```

Note that the first line wraps due to the size of the page, but it appears on one line in the SAS Enhanced Editor. The author indented only for readability. One may wish to add a line continuation if in doubt. This process requires substantial time and the author left it as an overnight process. The data far exceeded the 1 TB hard drive, so the author bought a Dell WD 4TB USB 3.0 WD My Book desktop external hard drive, which seems to work without meaningful issues.
CONCLUSION

This paper introduced the Entrez databases that the National Center for Biotechnology Information (NCBI) of the United States National Institute of Health (NIH) has made freely available and their programming interfaces that were incorporated into two SAS System® macros. The author provided a few examples of calls of the macros and briefly discussed some issues and results. The ultimate uses will be highly individualized or proprietary, so general and in-depth discussions are not immediately possible.

Certain issues concerning design need to be addressed. XML, for instance, should not be parsed as text. The DTD of the output should be stable enough to be worth the effort to use instead. Two calls to the M_U_EUTILS macro could be avoided with a design change and the names of the global macro variables can be prefixed with an incremented number so that the value is not overwritten or the need to capture it in another global macro variable after the call is not required, for instance, by checking the global macro table.

Programmatic approaches to such work have numerous advantages and the learning curve is not steep. Searches of a great number of items can be fast, but the macro has does not control the number of submissions, which can create an issue with NCBI. Contacting the help desk, which in the experience of the author has been quite responsive and helpful, is wise, especially if one might be submitting more than three submissions per second. SAS programs and their stored (permanent) logs document work and one may update search terms based on the results of executing programs.

REFERENCES

1 Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-.


3 Entrez Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2005-. Entrez Help. 2006 Jan 20 [Updated 2016 May 31].


CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kevin R. Viel, Ph.D.

Histonis, Incorporated
kviel@histonis.org

Navitas Data Sciences
kevin.viel@navitaslifesciences.com
www.navitasdatasciences.com

Any brand and product names are trademarks of their respective companies.
Appendix 1 The M_U_EUTILS macro.

```华北
%macro m_u_utilities
  e_utilities
    , db
    , term
    , extra_option
    , usehistory
    , api_key
    , query_key
    , webenv
    , results_out
    , out_ds
    , code
    , list_current_dbs
    , debug_url
    , debug_return
  /**************/
    , help
  )

%let e_utilities = %str()
%let db = %str()
%let term = %str()
%let extra_option = %str()
%let usehistory = %str()
%let api_key = %str()
%let query_key = %str()
%let webenv = %str()
%let results_out = %sysfunc(pathname(work))\results_out.xml
%let out_ds = %str()
%let code = %str()
%let list_current_dbs = N
%let debug_url = N
%let debug_return = N
%let help = N

%let mprint_orig = %sysfunc(getoption(mprint))

%if %mprint_orig = Y
%then
  %put %str(________________________________________________________________________________________);
  %put ________________________________________________________________________________________;
  %put ________________________________________________________________________________________;
  %let results_out = %sysfunc(pathname(work))\results_out.xml
  %let out_ds = %sysfunc(pathname(work))\out_ds.xml
  %let code = %sysfunc(pathname(work))\code.xml
  %let results_xml = %sysfunc(pathname(work))\results_xml.xml
  %let list_current_dbs = N
  %let debug_url = N
  %let debug_return = N
  %let help = N

%else

%put ________________________________________________________________________________________;
%put ________________________________________________________________________________________;
%put ________________________________________________________________________________________;
%let results_out = %sysfunc(pathname(work))\results_out.xml
%let out_ds = %sysfunc(pathname(work))\out_ds.xml
%let code = %sysfunc(pathname(work))\code.xml
%let results_xml = %sysfunc(pathname(work))\results_xml.xml
%let list_current_dbs = N
%let debug_url = N
%let debug_return = N
%let help = N

%end;
```
%let __rc = %sysfunc(fdelete(&temp.));
%let __rc = %sysfunc(filename(temp)); /* Delete the work xml file */
filename resp;
run;

proc http
method = "GET"
out = resp
method = "GET"
run;

data %if &list_current_dbs. ne Y
%then __current_dbs
  ( keep = eutility_database_name )
%else_null_;
infile resp;
input;
length eutility_database_name $ 50;
__rc = prxparse("/<DbName>({a-zA-Z0-9})*</DbName>/")
retain __rc;
if prxmatch(__rc, _infile_) then do;
eutility_database_name = prxposn(__rc
1, _infile_);
output;
%if __current_dbs. = Y
%then put eutility_database_name %str();
end;
run;

filename resp;
/* Delete the work xml file */
%let __rc = %sysfunc(filename(temp)
%sysfunc(pathname(work))/work_tmp.xml)
%sysfunc(fdelete(&temp.));
%if &list_current_dbs. = Y
  %then
    %do;
    %goto __END;
    %end;
  
/*******
%if %sysfunc(prxmatch(/^(?:(einfo.fcgi|esearch.fcgi|epost.fcgi|esummary.fcgi|efetch.fcgi|elink.fcgi|egquery.fcgi|espell.fcgi|ecitmatch.cgi)$/i , &e_utilities. )) = 0
  %then
    %do;
    %put E%str(RROR: ) E_UTILITIES = &e_utilities. must be one of:
    %put E%str(RROR: ) einfo.fcgi    ;
    %put E%str(RROR: ) esearch.fcgi  ;
    %put E%str(RROR: ) epost.fcgi    ;
    %put E%str(RROR: ) esummary.fcgi ;
    %put E%str(RROR: ) efetch.fcgi   ;
    %put E%str(RROR: ) elink.fcgi    ;
    %put E%str(RROR: ) egquery.fcgi  ;
    %put E%str(RROR: ) espell.fcgi   ;
    %put E%str(RROR: ) ecitmatch.cgi ;
    %goto __END;
    %end;
  
  /**********/
%if &db. ne %str()
  %then
    %do;
    proc sql
      noprint;
      select eutility_database_name
      from __current_dbs
      where eutility_database_name = "&db."
    ;
    quit;
    %if &sqlobs. = 0
    %then
      %do;
      %put E%str(RROR: ) DB = &db. must be one of (case specific): ;
      data _null_;  
      set __current_dbs;
      put "E" "RROR: " eutility_database_name ;
      run ;
      %goto __END;
      %end;
    %end;
  %end;
  
  /************************************************************/
%if %nrbquote(&db.) ne %str()
  %then  %let url = &url.?db=&db. ;
%if %nrbquote(&term.) ne %str()
  %then  %let url = &url.%nrstr(&)term=&term. ;
%if %nrbquote(&query_key.) ne %str()
  %then  %let url = &url.%nrstr(&)query_key=&query_key. ;
%if %nrbquote(&webenv.) ne %str()
  %then  %let url = &url.%nrstr(&)WebEnv=&webenv. ;
%if %nrbquote(&usehistory.) ne %str()
  %then
    %do ;
%global querykey_return
webenv_return

%let url = &url.%nrstr(&)usehistory=&usehistory.

;if %nrbquote(&api_key.) ne %str()
%then %let url = &url.%nrstr(&)api_key=&api_key.
;if &debug_url. = Y
%then

/***/
filename resp
"&results_out."
;
proc http
url = %unquote(%str(%')&url.%str(%'))
out = resp
method = "GET"
;
run;
data
%if %nrbquote(&out_ds.) = %str() %then _null_
%else &out_ds.
;
infile resp;
input;
%if &debug_return. = Y %then put _infile_ %str();
&code.
run;
filename resp;
%__END:
skip;
%put ************************************************************************************************************************
%put Please see: https://www.ncbi.nlm.nih.gov/home/about/policies/
%put Disclaimer and Copyright Issues
%put If you use the Utilities within software, NCBI's Disclaimer and Copyright notice (https://www.ncbi.nlm.nih.gov/About/disclaimer.html) must be evident to users of your product. Please note that abstracts in PubMed may incorporate material that may be protected by U.S. and foreign copyright laws. All persons reproducing, redistributing, or making commercial use of this information are expected to adhere to the terms and conditions asserted by the copyright holder. Transmission or reproduction of protected items beyond that allowed by fair use (PDF) as defined in the copyright laws requires the written permission of the copyright owners.
%put **************************************************************************************************************************************************
%put ____________________________________________________________________________________________________________________________________________________
%put End of help
%mend m_u_eutils;

Appendix 2 The M_NCBI_CLI_TOOL macro.

%macro m_ncbi_cli_tool
(cli_tool          = datasets,
   cli_tool_path     = ~\datasets.exe,
   api_key           = %str(),
   cli_tool_help     = N,
   options           = %str(),
   pre               = %str(),
   records_to_read   = %str(),
);
%if &help. = Y
%then
%do;

%let mprint_orig = %sysfunc(getoption(mprint));
%options nomprint;
skip;
skip;

%put ________________________________________________________________________________________________________________________
_______________;
%put Purpose of program: This macro runs the NCBI Datasets Command line interface (CLI) tool reference
guide                                     ;
%put %str(                     )The NCBI Datasets datasets command line tools include datasets and dataformat. They can be us
ed to          ;
%put %str(                     )download and convert metadata into tabular format.                                                          ;
%put %str(                     )See: https://www.ncbi.nlm.nih.gov/datasets/docs/v1/reference-
command-line/                             ;
%put _________________    ___________________________________________________________________
______________________________________________ ;
%put cli_tool           = The CLI tool to use: datasets or dataformat. ;
%put %str(                        )Default: datasets                                                                                        ;
%put cli_tool_path      = The path to executable (.exe) files. ;
%put %str(                        )Default: %nrstr(%%)str%str(()                                                                           ;
%put api_key            = NCBI API key for the user. ;
%put %str(                     )See:  https://support.nlm.nih.gov/knowledgebase/article/KA-
05317/en                                      ;
%put %str(                        )Default: %nrstr(%%)str%str(())                                                                           ;
%put cli_tool_help      = Whether to display the tool help, this is not the macro help. ;
%put options
= Free-text options to add to the options of the command line.                                                      ;
%put %str(                        )Default: %nrstr(%%)str%str(()                                                                           ;
%put pre                = Free-text command line to add before the tool. For instance, this can be used to change directory. ;
%put %str(                        )Default: %nrstr(%%)str%str(()                                                                           ;
%put records_to_read    = A number value of the number of records to read in the pipe results. ;
%put %str(                        )Default: %nrstr(%%)str%str(())                                                                           ;

%put __END:_________________________;_________________________;_________________________;_________________________;_________________________;_________________________;_________________________; ______________________;
%options &mprint_orig.;
%goto __END ;
%end;

%if &cli_tool_help. = Y
%then
%do;

filename cli_tool
pipe
  %unquote(%str($)"&cli_tool_path. "-h%str($))
;  

data_null ;
  infile cli_tool ;
  input ;
  put _infile_ ;
  run ;

filename cli_tool ;
%g0to __END ;
%end ;

/**************************
%if %nrbquote(&api_key.) ne %str() %then %let
options = &options --api-key &api_key. ;
**************************

filename cli_tool
pipe
  %unquote(%str($) &pre. "&cli_tool_path. " &options. %str($))
;  

data_null ;
  infile cli_tool ;
  input ;
  put _ infile_ ;
  %if %sysfunc( prxmatch( /\d+$/ , %nrbquote(&records_to_read.))
  and &records_to_read. > 0
  %then if _n_ > &records_to_read. then stop %str($); 
  run ;

filename cli_tool ;
%g0to _END;
%end;
%end:
%mend  m_ncbi_cli_tool ;