Entrez Direct Reference

Searching, Retrieving, and Parsing Data from NCBI Databases through the Unix Command Line

Table of Contents

Introduction

Entrez Direct (EDirect) provides access to the NCBI's suite of interconnected databases from a Unix terminal window. Search terms are entered as command-line arguments. Individual operations are connected with Unix pipes to allow construction of multi-step queries. Selected records can then be retrieved in a variety of formats.

EDirect also includes an argument-driven utility that simplifies the extraction of results in structured XML or JSON format, and a program that builds a URL from command-line arguments for easy access to external CGI data services. These can eliminate the need for writing custom software to answer ad hoc questions.

Queries can move seamlessly between EDirect programs and Unix utilities or scripts to perform actions that cannot be accomplished entirely within Entrez.

Programmatic Access

EDirect connects to Entrez through the Entrez Programming Utilities interface. It supports searching by indexed terms, looking up precomputed neighbors or links, filtering results by date or category, and downloading record summaries or reports.

EDirect navigation programs (**esearch**, **elink**, **efilter**, and **efetch**) communicate by means of a small structured message, which can be passed invisibly between operations with a Unix pipe. The message includes the current database, so it does not need to be given as an argument after the first step.

All EDirect programs are designed to work on large sets of data. Intermediate results are stored on the Entrez history server. For best performance, obtain an API Key from NCBI, and place the following line in your .bash_profile configuration file:

```
export NCBI_API_KEY=unique_api_key_goes_here
```

Each program also has a **-help** command that prints detailed information about available arguments.

Navigation Functions

Esearch performs a new Entrez search using terms in indexed fields. It requires a **-db** argument for the database name and uses **-query** to obtain the search terms. For PubMed, without field qualifiers, the server uses automatic term mapping to compose a search strategy by translating the supplied query:

```
esearch -db pubmed -query "selective serotonin reuptake inhibitor"
```

Search terms can also be qualified with bracketed field names:

```
esearch -db nucleotide -query "insulin [PROT] AND rodents [ORGN]"
```

Elink looks up precomputed neighbors within a database, or finds associated records in other databases:

```
elink -related
```

```
elink -target gene
```

or can follow PubMed references in the NIH Open Citation Collection dataset (see PMID 31600197):

```
elink -cited

elink -cites
```

Efilter limits the results of a previous query, with shortcuts that can also be used in esearch:

```
efilter -molecule genomic -location chloroplast -country sweden -days 365
```

Efetch downloads selected records or reports in a designated format:

```
efetch -format abstract
```

Individual query commands are connected by a Unix vertical bar pipe symbol:

```
esearch -db pubmed -query "tn3 transposition immunity" | efetch -format medline
```

Discovery by Navigation

PubMed related articles are calculated by a statistical text retrieval algorithm using the title, abstract, and medical subject headings (MeSH terms). The connections between papers can be used for making discoveries. A simple example is finding the last enzymatic step in the vitamin A biosynthetic pathway.

Lycopene cyclase in plants converts lycopene to β-carotene, the immediate precursor of vitamin A. An initial search on the enzyme finds 246 articles. Looking up precomputed neighbors:

```
esearch -db pubmed -query "lycopene cyclase" |
elink -related |
```

returns 14,958 PubMed papers, some of which might be expected to discuss adjacent steps in the pathway. Since plants cannot convert β-carotene to retinal, we first link to proteins, finding 391,755 sequence records (each of which has standardized organism information from the NCBI taxonomy). Next we restrict those results to animals, to eliminate earlier steps in the pathway. Limiting to curated proteins in mice matches 25 records:

```
elink -target protein |
efilter -organism mouse -source refseq |
```

This is small enough to examine individually, so we retrieve the records in FASTA format:

```
efetch -format fasta
```

As anticipated, the results include the enzyme that splits β-carotene into two molecules of retinal:

```
...
>NP_067461.2 beta,beta-carotene 15,15'-dioxygenase isoform 1 [Mus musculus]
MEIIFGQNKKEQLEPVQAKVTGSIPAWLQGTLLRNGPGMHTVGESKYNHWFDGLALLHSFSIRDGEVFYR
SKYLQSDTYIANIEANRIVVSEFGTMAYPDPCKNIFSKAFSYLSHTIPDFTDNCLINIMKCGEDFYATTE
TNYIRKIDPQTLETLEKVDYRKYVAVNLATSHPHYDEAGNVLNMGTSVVDKGRTKYVIFKIPATVPDSKK
...
```

The entire set of commands runs in 8 seconds. There is no need to use a script to loop over records one at a time, or write code to retry after a transient network failure, or add a time delay between requests. All of these features are already built into the EDirect commands.

XML Data Extraction

The ability to obtain Entrez records in structured format, and to easily extract the underlying data, allows the user to ask novel questions that are not addressed by existing analysis software.

The **xtract** program uses command-line arguments to direct the conversion of XML data into a more tractable form. The **-pattern** command partitions an XML stream into individual records that are processed separately. Within each record, the **-element** command does an exhaustive, depth-first search to find data content by field name. Explicit paths to objects are not needed.

Selection commands are derivatives of -element. These include positional commands (**-first** and **-last**), numeric operations (including **-num**, **-len**, **-inc**, **-sum**, **-min**, **-max**, and **-avg**), text processing variants (such as **-encode**, **-plain**, **-upper**, **-title**, and **-words**), and functions that perform sequence or coordinate conversion (**-revcomp**, **-0-based**, **-1-based**, and **-ucsc-based**).

Format Customization

By default, the -pattern argument divides the results into rows, while placement of data into columns is controlled by -element, to create a tab-delimited table.

Formatting commands allow extensive customization of the output. The line break between -pattern output rows can be changed with **-ret**, and the tab character between -element fields can be replaced by **-tab**. The **-sep** argument is used to distinguish multiple elements of the same type, and controls their separation independently of the -tab command. The following query:

```
efetch -db pubmed -id 6271474,6092233,16589597 -format docsum |
xtract -pattern DocumentSummary -sep "|" -element Id PubDate Name
```

returns a table with individual author names separated by vertical bars:

```
6271474     1981            Casadaban MJ|Chou J|Lemaux P|Tu CP|Cohen SN
6092233     1984 Jul-Aug    Calderon IL|Contopoulou CR|Mortimer RK
16589597    1954 Dec        Garber ED
```

The **-def** command sets a default placeholder to be printed when an -element field is not present.

Exploration Control

Exploration commands provide fine control over the order in which XML record contents are examined, by presenting each instance of a selected subregion separately. This limits what subsequent commands "see", and allows related fields in an object to be kept together.

In contrast to the simpler DocumentSummary format, records retrieved as PubmedArticle XML:

```
efetch -db pubmed -id 1413997 -format xml |
```

have authors with separate fields for last name and initials:

```
<Author>
  <LastName>Mortimer</LastName>
  <Initials>RK</Initials>
</Author>
```

Without being given any guidance about context, an -element command on initials and last names:

```
xtract -pattern PubmedArticle -element Initials LastName
```

will explore the current record for each argument in turn, and thus print all author initials followed by all author last names:

```
RK    CR    JS    Mortimer    Contopoulou    King
```

Inserting a **-block** command redirects data exploration to present each author one at a time. The subsequent -element command only sees the current author's values:

```
xtract -pattern PubmedArticle -block Author -element Initials LastName
```

which restores the correct association of initials and last names:

```
RK    Mortimer    CR    Contopoulou    JS    King
```

The -sep value also applies to unrelated -element arguments that are grouped with commas:

```
xtract -pattern PubmedArticle \
  -block Author -sep " " -tab ", " -element Initials,LastName
```

allowing -sep -and -tab to produce a more desirable formatting of author names:

```
RK Mortimer, CR Contopoulou, JS King
```

Nested Exploration

Exploration command names (**-group**, **-block**, and **-subset**) are assigned to a precedence hierarchy:

```
-pattern > -group > -block > -subset > -element
```

and are combined in ranked order to control object iteration at progressively deeper levels in the XML data structure. Each command argument acts as a "nested for-loop" control variable, retaining information about the context, or state of exploration, at its level.

(Hypothetical) census data would need several nested loops to visit each unique address in context:

```
-pattern State -group City -block Street -subset Number -element Resident
```

MeSH terms can have their own unique set of qualifiers, with a major topic attribute on each object:

```
<MeshHeading>
  <DescriptorName MajorTopicYN="N">beta-Galactosidase</DescriptorName>
  <QualifierName MajorTopicYN="Y">genetics</QualifierName>
  <QualifierName MajorTopicYN="N">metabolism</QualifierName>
</MeshHeading>
```

Since -element does its own exploration for objects within its current scope, a -block command:

```
-block MeshHeading -sep " / " -element DescriptorName,QualifierName
```

is sufficient for grouping each MeSH name with its qualifiers:

```
beta-Galactosidase / genetics / metabolism
```

Adding **-subset** commands within the -block visits each individual descriptor or qualifier object on the current MeSH term:

```
efetch -db pubmed -id 6162838 -format xml |
xtract -transform <( echo -e "Y\t*\n" ) \
  -pattern PubmedArticle -element MedlineCitation/PMID \
    -block MeshHeading -clr \
      -subset DescriptorName -plg "\n" -tab "" \
        -translate "@MajorTopicYN" -element DescriptorName \
      -subset QualifierName -plg " / " -tab "" \
        -translate "@MajorTopicYN" -element QualifierName
```

and keeps major topic attributes associated with their parent objects. A text translation command converts the "Y" attribute value to an asterisk for printing:

```
6162838
Base Sequence
*DNA, Recombinant
Escherichia coli / genetics
...
RNA, Messenger / *genetics
Transcription, Genetic
beta-Galactosidase / *genetics / metabolism
```

(Note that "-element MedlineCitation/PMID" uses the **parent-slash-child** construct to prevent the display of additional PMID items that may be present later in CommentsCorrections objects.)

Conditional Execution

Conditional processing commands (**-if**, **-unless**, **-and**, **-or**, and **-else**) restrict exploration by object name and value. These may be used in conjunction with string or numeric constraints:

```
esearch -db pubmed -query "Casadaban MJ [AUTH]" |
efetch -format xml |
xtract -pattern PubmedArticle -if "#Author" -lt 6 \
  -block Author -if LastName -is-not Casadaban \
    -sep ", " -tab "\n" -element LastName,Initials |
sort-uniq-count-rank
```

to select papers with fewer than 6 authors and print a table of the most frequent coauthors:

```
11    Chou, J
8     Cohen, SN
7     Groisman, EA
...
```

Saving Data in Variables

A value can be recorded in a variable and used wherever needed. Variables are created by a hyphen followed by a name consisting of a string of capital letters or digits (e.g., **-PMID**). Values are retrieved by placing an ampersand before the variable name (e.g., "**&PMID**") in an -element statement:

```
efetch -db pubmed -id 3201829,6301692,781293 -format xml |
xtract -pattern PubmedArticle -PMID MedlineCitation/PMID \
  -block Author -element "&PMID" \
    -sep " " -tab "\n" -element Initials,LastName
```

producing a list of authors, with the PubMed Identifier in the first column of each row:

```
3201829    JR Johnston
3201829    CR Contopoulou
3201829    RK Mortimer
6301692    MA Krasnow
6301692    NR Cozzarelli
781293     MJ Casadaban
```

The variable can be used even though the original object is no longer visible inside the -block section.

Sequence Qualifiers

The NCBI represents sequence records in a data model based on the central dogma of molecular biology. A sequence can have multiple features, which contain information about the biology of a given region, including the transformations involved in gene expression. Each feature can have multiple qualifiers, which store specific details about that feature (e.g., name of the gene, genetic code used for translation, accession of the product sequence).

The data hierarchy is explored using a **-pattern** {sequence} **-group** {feature} **-block** {qualifier} construct. As a convenience, an **-insd** helper function generates the appropriate nested extraction commands from feature and qualifier names on the command line. For example, processing the results of a search on cone snail venom:

```
esearch -db protein -query "conotoxin" -feature mat_peptide |
efetch -format gpc |
xtract -insd complete mat_peptide "%peptide" product mol_wt peptide |
grep -i conotoxin | sort -t $'\t' -u -k 2,2n
```

returns the accession, peptide length, product name, calculated molecular weight, and sequence for a sample of neurotoxic peptides:

```
ADB43131.1    15    conotoxin Cal 1b      1708    LCCKRHHGCHPCGRT
ADB43128.1    16    conotoxin Cal 5.1     1829    DPAPCCQHPIETCCRR
AIC77105.1    17    conotoxin Lt1.4       1705    GCCSHPACDVNNPDICG
ADB43129.1    18    conotoxin Cal 5.2     2008    MIQRSQCCAVKKNCCHVG
ADD97803.1    20    conotoxin Cal 1.2     2206    AGCCPTIMYKTGACRTNRCR
AIC77085.1    21    conotoxin Bt14.8      2574    NECDNCMRSFCSMIYEKCRLK
ADB43125.1    22    conotoxin Cal 14.2    2157    GCPADCPNTCDSSNKCSPGFPG
AIC77154.1    23    conotoxin Bt14.19     2578    VREKDCPPHPVPGMHKCVCLKTC
...
```

Genes in a Region

To list all genes between two markers flanking the human X chromosome centromere, first retrieve the chromosome record:

```
esearch -db gene -query "Homo sapiens [ORGN] AND X [CHR]" |
efilter -status alive -type coding | efetch -format docsum |
```

Gene names and chromosomal positions are extracted by piping the record to:

```
xtract -pattern DocumentSummary -NME Name -DSC Description \
  -block GenomicInfoType -if ChrLoc -equals X \
    -min ChrStart,ChrStop -element "&NME" "&DSC" |
```

Exploring each GenomicInfoType is needed because of pseudoautosomal regions at the ends of the X and Y chromosomes. Without limiting to chromosome X, the copy of IL9R near the "q" telomere of chromosome Y would be erroneously placed with genes that are near the X chromosome centromere.

Results can now be sorted, filtered, and passed to the between-two-genes script:

```
sort -k 1,1n | cut -f 2- |
grep -v pseudogene | grep -v uncharacterized |
between-two-genes AMER1 FAAH2
```

to produce a table of known genes located between the two markers:

```
FAAH2      fatty acid amide hydrolase 2
SPIN2A     spindlin family member 2A
ZXDB       zinc finger X-linked duplicated B
NLRP2B     NLR family pyrin domain containing 2B
ZXDA       zinc finger X-linked duplicated A
SPIN4      spindlin family member 4
ARHGEF9    Cdc42 guanine nucleotide exchange factor 9
AMER1      APC membrane recruitment protein 1
```

Genes in a Pathway

A gene can be linked to the biochemical pathways in which it participates:

```
esearch -db gene -query "PAH [GENE]" -organism human |
elink -target biosystems |
efilter -pathway wikipathways |
```

Linking from a pathway record back to the gene database:

```
elink -target gene |
efetch -format docsum |
xtract -pattern DocumentSummary -element Name Description |
grep -v pseudogene | grep -v uncharacterized |
sort -f
```

returns the set of all genes known to be involved in the pathway:

```
AANAT      aralkylamine N-acetyltransferase
ACADM      acyl-CoA dehydrogenase medium chain
ACHE       acetylcholinesterase (Cartwright blood group)
ADCYAP1    adenylate cyclase activating polypeptide 1
...
```

Gene Sequence

Genes encoded on the minus strand of a sequence:

```
esearch -db gene -query "DDT [GENE] AND mouse [ORGN]" |
efetch -format docsum |
xtract -pattern GenomicInfoType -element ChrAccVer ChrStart ChrStop |
```

have coordinates where the start position is greater than the stop:

```
NC_000076.6    75773373    75771232
```

These can be read by a "while" loop:

```
while IFS=$'\t' read acn str stp
do
  efetch -db nucleotide -format gb \
    -id "$acn" -chr_start "$str" -chr_stop "$stp"
done
```

to return the reverse-complemented subregion in GenBank format:

```
LOCUS       NC_000076               2142 bp    DNA     linear   CON 08-AUG-2019
DEFINITION  Mus musculus strain C57BL/6J chromosome 10, GRCm38.p6 C57BL/6J.
ACCESSION   NC_000076 REGION: complement(75771233..75773374)
VERSION     NC_000076.6
...
FEATURES             Location/Qualifiers
     source          1..2142
                     /organism="Mus musculus"
                     /mol_type="genomic DNA"
                     /strain="C57BL/6J"
                     /db_xref="taxon:10090"
                     /chromosome="10"
     gene            1..2142
                     /gene="Ddt"
     mRNA            join(1..159,462..637,1869..2142)
                     /gene="Ddt"
                     /product="D-dopachrome tautomerase"
                     /transcript_id="NM_010027.1"
     CDS             join(52..159,462..637,1869..1941)
                     /gene="Ddt"
                     /codon_start=1
                     /product="D-dopachrome decarboxylase"
                     /protein_id="NP_034157.1"
                     /translation="MPFVELETNLPASRIPAGLENRLCAATATILDKPEDRVSVTIRP
                     GMTLLMNKSTEPCAHLLVSSIGVVGTAEQNRTHSASFFKFLTEELSLDQDRIVIRFFP
                     ...
```

The reverse complement of a plus-strand sequence range can be selected with efetch -revcomp.

Recursive Definitions

When a recursively defined object is given to an exploration command:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml |
xtract -pattern Taxon -element TaxId ScientificName
```

the -element command only examines fields in the outermost objects:

```
9606     Homo sapiens
7227     Drosophila melanogaster
10090    Mus musculus
```

The **star-slash-child** construct will descend a single level into the hierarchy:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml |
xtract -pattern Taxon -block "*/Taxon" \
  -if Rank -is-not "no rank" \
    -tab "\n" -element TaxId,Rank,ScientificName
```

to print data on the individual lineage objects:

```
2759     superkingdom    Eukaryota
33208    kingdom         Metazoa
7711     phylum          Chordata
89593    subphylum       Craniata
8287     superclass      Sarcopterygii
40674    class           Mammalia
...
```

Recursive objects can be fully explored with a **double-star-slash-child** construct:

```
esearch -db gene -query "rbcL [GENE] AND maize [ORGN]" |
efetch -format xml |
xtract -pattern Entrezgene -block "**/Gene-commentary" \
```

Metadata annotated in an attribute:

```
<Gene-commentary_type value="genomic">1</Gene-commentary_type>
```

is selected with an "at" sign before the attribute name:

```
  -if Gene-commentary_type@value -equals genomic \
    -tab "\n" -element Gene-commentary_accession |
sort | uniq
```

This prints every genomic accession regardless of nesting depth:

```
NC_001666
X86563
Z11973
```

Heterogeneous Objects

The **nquire** program uses command-line arguments to request data from external CGI services. A query on curated biological database associations:

```
nquire -get http://mygene.info/v3/gene/2652 |
xtract -j2x -set - -rec GeneRec |
```

returns data containing a heterogeneous mixture of objects in the pathway section:

```
<pathway>
  <reactome>
    <id>R-HSA-162582</id>
    <name>Signal Transduction</name>
  </reactome>
  ...
  <wikipathways>
    <id>WP455</id>
    <name>GPCRs, Class A Rhodopsin-like</name>
  </wikipathways>
</pathway>
```

The **parent-slash-star** construct is used to visit the individual components of a parent object without needing to explicitly specify their names. For printing, the name of a child object is indicated by a question mark:

```
xtract -pattern GeneRec -group "pathway/*" \
  -pfc "\n" -element "?,name,id"
```

This displays a table of pathway database references:

```
reactome        Signal Transduction                R-HSA-162582
reactome        Disease                            R-HSA-1643685
...
reactome        Diseases of signal transduction    R-HSA-5663202
wikipathways    GPCRs, Class A Rhodopsin-like      WP455
```

Indexed Fields

Entrez can report the fields and links that are indexed for each database. For example:

```
einfo -db protein -fields
```

will return a table of field abbreviations and names indexed for proteins:

```
ACCN    Accession
ALL     All Fields
ASSM    Assembly
AUTH    Author
BRD     Breed
CULT    Cultivar
DIV     Division
ECNO    EC/RN Number
FILT    Filter
FKEY    Feature key
GENE    Gene Name
...
```

Local PubMed Cache

Fetching data from Entrez works well when a few thousand records are needed, but it does not scale for much larger sets of data, where the time it takes to download becomes a limiting factor. EDirect can now preload all 30 million PubMed records onto an inexpensive external 500 GB solid state drive for rapid retrieval.

For example, PMID 12345678 would be stored (as a compressed XML file) at:

```
/Archive/12/34/56/12345678.xml.gz
```

using a hierarchy of folders to organize the data for random access to any record.

Set an environment variable in your .bash_profile configuration file to reference your external drive:

```
export EDIRECT_PUBMED_MASTER=/Volumes/external_disk_name_goes_here
```

and run:

```
archive-pubmed
```

to download the PubMed release files and distribute each record on the drive. This process will take several hours to complete, but subsequent updates are incremental, and should finish in minutes.

The local archive is a completely self-contained, turnkey system, with no need for the user to download and configure complicated third-party database software.

Retrieving a PubmedArticleSet containing over 120,000 PubMed records from the local archive:

```
esearch -db pubmed -query "PNAS [JOUR]" -pub abstract |
efetch -format uid | stream-pubmed | gunzip -c |
```

takes about 15 seconds. Retrieving those records from NCBI's network service, with efetch -format xml, would take around 40 minutes.

Even moderately large sets of PubMed query results can benefit from using the local cache. A reverse citation lookup on 191 papers:

```
esearch -db pubmed -query "Cozzarelli NR [AUTH]" | elink -cited |
```

requires 5 seconds to match 7156 subsequent articles. Fetching them from the local archive:

```
efetch -format uid | fetch-pubmed |
```

is practically instantaneous. Printing the names of all authors in those records:

```
xtract -pattern PubmedArticle -block Author \
  -sep " " -tab "\n" -element LastName,Initials |
```

allows creation of a frequency table:

```
sort-uniq-count-rank
```

that lists the authors who most often cited the original papers:

```
112    Cozzarelli NR
73     Maxwell A
56     Wang JC
49     Osheroff N
48     Stasiak A
...
```

Fetching from the network service would extend the 7 second running time by 2 minutes.

Local Search Index

A similar divide-and-conquer strategy is used to create a local information retrieval system suitable for large data mining queries. Run:

```
index-pubmed
```

to populate retrieval index files from records stored in the local archive. This will also take a few hours.

For PubMed titles and primary abstracts, the indexing process deletes hyphens after specific prefixes, removes accents and diacritical marks, splits words at punctuation characters, corrects encoding artifacts, and spells out Greek letters for easier searching on scientific terms. It then prepares inverted indices with term positions, and uses them to build distributed term lists and postings files.

For example, the term list that includes "cancer" would be located at:

```
/Postings/NORM/c/a/n/c/canc.trm
```

A query on cancer thus only needs to load a very small subset of the total index. This design allows efficient expression evaluation, unrestricted wildcard truncation, phrase queries, and proximity searches.

The **phrase-search** script provides access to the local search system. The full set of indexed terms, without record counts, can be printed for any field:

```
phrase-search -terms NORM
```

In local queries, a trailing asterisk is used to indicate term truncation:

```
phrase-search -count "catabolite repress*"
```

Using -counts returns expanded terms and individual postings counts:

```
phrase-search -counts "catabolite repress*"
```

Query evaluation includes Boolean operations and parenthetical expressions:

```
phrase-search -query "(literacy AND numeracy) NOT (adolescent OR child)"
```

Adjacent words in the query are treated as a contiguous phrase:

```
phrase-search -query "selective serotonin reuptake inhibit*"
```

More inclusive searches can use the Porter2 stemming algorithm:

```
phrase-search -query "monoamine oxidase inhibitor [STEM]"
```

Each plus sign will replace a single word inside a phrase:

```
phrase-search -query "vitamin c + + common cold"
```

Runs of tildes indicate the maximum distance between phrases:

```
phrase-search -query "vitamin c ~ ~ common cold"
```

MeSH hierarchy code and year of publication are also indexed:

```
phrase-search -query "C14.907.617.812* [TREE] AND 2015:2019 [YEAR]"
```

An exact match can search for all or part of a title or abstract:

```
phrase-search -exact "Genetic Control of Biochemical Reactions in Neurospora."
```

All query commands return a list of PMIDs, which can be piped directly to fetch-pubmed to retrieve the records. For example:

```
phrase-search -query "selective serotonin ~ ~ ~ reuptake inhibitor*" |
fetch-pubmed |
xtract -pattern PubmedArticle -num Author |
sort-uniq-count -n |
reorder-columns 2 1 |
head -n 25 |
tee /dev/tty |
xy-plot auth.png
```

performs a proximity search with dynamic wildcard expansion (matching phrases like "selective serotonin and norepinephrine reuptake inhibitors") and fetches 12,170 PubMed records from the local archive. It then counts the number of authors for each paper, printing a frequency table of the number of papers per number of coauthors:

```
0    49
1    1350
2    1827
3    1835
4    1661
5    1457
6    1133
7    907
8    597
9    408
...
```

and creating a visual graph of the data. The entire set of commands runs in under 4 seconds.

The phrase-search and fetch-pubmed scripts are front-ends to the **rchive** program, which is used to build and search the inverted retrieval system. Rchive is multi-threaded for speed, and can match several PubMed titles per second, fetching the positional indices for all terms in parallel before evaluating the title words as a contiguous phrase.

Rapidly Scanning PubMed

If the **expand-current** script is run after archive-pubmed or index-pubmed, an ad hoc scan can be performed on the entire set of live PubMed records:

```
cat $EDIRECT_PUBMED_MASTER/Current/*.xml |
xtract -timer -pattern PubmedArticle \
  -if "#Author" -eq 7 \
    -element MedlineCitation/PMID LastName
```

in this case finding articles with seven authors. (Author count is not indexed by Entrez or locally by EDirect.)

Xtract uses the Boyer-Moore-Horspool algorithm to partition an XML stream into individual records, sending them down a thread-safe communication channel to be distributed among multiple instances of the data exploration and extraction function. On a modern six-core computer, it can process the full scan of all 30 million PubMed records in just under 4 minutes, a sustained rate of over 125,000 records per second.

Identifier Conversion

The index-pubmed script also downloads MeSH descriptor information from NLM and creates a conversion file:

```
...
<Rec>
  <Code>D064007</Code>
  <Name>Ataxia Telangiectasia Mutated Proteins</Name>
  <Tree>D08.811.913.696.620.682.700.097</Tree>
  <Tree>D12.776.157.687.125</Tree>
  <Tree>D12.776.660.720.125</Tree>
</Rec>
...
```

that can be used for mapping MeSH codes to and from chemical or disease names. For example:

```
cat $EDIRECT_PUBMED_MASTER/Data/meshconv.xml |
xtract -pattern Rec \
  -if Name -starts-with "ataxia telangiectasia" \
    -element Code
```

will return:

```
C565779
C576887
D001260
D064007
```

The meshconv.xml file is prepared by use of the xtract **-wrp** command:

```
cat desc2020.xml |
xtract -wrp Set,Rec -pattern DescriptorRecord \
  -wrp Code -element DescriptorRecord/DescriptorUI \
  -wrp Name -first DescriptorName/String \
  -wrp Tree -element TreeNumberList/TreeNumber |
xtract -format |
xtract -wrp Set -pattern Rec -sort Code
```

which wraps element contents in new XML tags by issuing several other formatting commands:

```
-pfx "<Tree>" -sep "</Tree><Tree>" -sfx "</Tree>"
```

Natural Language Processing

Additional annotation on PubMed can be downloaded and indexed by running:

```
index-extras
```

NCBI's Biomedical Text Mining Group performs computational analysis of PubMed and PMC papers, and extracts chemical, disease, and gene references from the article contents (see PMID 31114887). Along with NLM Gene Reference Into Function mappings (see PMID 14728215), these terms are indexed in CHEM, DISZ, and GENE fields.

Recent research at Stanford defined biological themes, supported by dependency paths, which are indexed as THME and PATH fields. Theme keys in the Global Network of Biomedical Relationships are taken from a table in the paper (see PMID 29490008):

A+	Agonism, activation	**N**	Inhibits
A-	Antagonism, blocking	**O**	Transport, channels
B	Binding, ligand	**Pa**	Alleviates, reduces
C	Inhibits cell growth	**Pr**	Prevents, suppresses
D	Drug targets	**Q**	Production by cell population
E	Affects expression/production	**Rg**	Regulation
E+	Increases expression/production	**Sa**	Side effect/adverse event
E-	Decreases expression/production	**T**	Treatment/therapy
G	Promotes progression	**Te**	Possible therapeutic effect
H	Same protein or complex	**U**	Causal mutations
I	Signaling pathway	**Ud**	Mutations affecting disease course
J	Role in disease pathogenesis	**V+**	Activates, stimulates
K	Metabolism, pharmacokinetics	**W**	Enhances response
L	Improper regulation linked to disease	**X**	Overexpression in disease
Md	Biomarkers (diagnostic)	**Y**	Polymorphisms alter risk
Mp	Biomarkers (progression)	**Z**	Enzyme activity

Themes common to multiple chemical-disease-gene relationships are disambiguated so they can be queried individually. The expanded list, along with MeSH category codes and examples of query automation, can be seen with:

```
phrase-search -help
```

Integration with Entrez

The phrase-search **-filter** command allows PMIDs to be generated by an EDirect search and then incorporated as a component in a local query:

```
esearch -db pubmed -query "complement system proteins [MESH]" |
efetch -format uid |
phrase-search -filter "L [THME] AND D10* [TREE]"
```

This finds PubMed papers about complement proteins and limits them by the "improper regulation linked to disease" theme and the lipids MeSH chemical category:

```
448084
1292783
1379443
1467432
1689670
...
```

Intermediate lists of PMIDs can be saved to a file and piped (with "cat") into a subsequent phrase-search -filter query, or uploaded to the Entrez history server by piping to:

```
epost -db pubmed -format uid
```

External Services

The experimental **xplore** script expands the EDirect paradigm to navigate connections in the biological resources of the BioThings.io data integration project at Scripps Research (see PMID 23175613). A drug repurposing example (see PMID 29390967):

```
xplore -load hgvs "chr6:g.26093141G>A,chr12:g.111351981C>T" |
xplore -link ncbigene |
xplore -link wikipathways |
xplore -link ncbigene |
xplore -link uniprot |
xplore -link inchikey |
xplore -save uid
```

runs in 20 seconds and returns 1042 chemicals that might act on gene products in pathways associated with two diseases, and would thus be potential candidates for treating hereditary hemochromatosis or hypertrophic cardiomyopathy. There is initial support in xplore -search for -organism and -action shortcuts, similar to what is available in efilter for Entrez data.

As part of this development, xtract gained a **-path** exploration command and support for multi-level object addresses, delimited by periods or slashes:

```
xtract -path pathway.wikipathways.id -tab "\n" -element id
```

JSON to XML

Consolidated gene information retrieved in JSON format:

```
nquire -get http://mygene.info/v3 gene 3043 |
```

contains a multi-dimensional JSON array of exon coordinates:

```
"position": [
  [
    5225463,
    5225726
  ],
  [
    5226576,
    5226799
  ],
  [
    5226929,
    5227071
  ]
],
```

This can be converted to XML with xtract **-j2x**:

```
xtract -j2x -set - -rec GeneRec -nest plural |
```

using "-nest plural" to derive a parent name that keeps the internal structure intact in XML:

```
<positions>
  <position>5225463</position>
  <position>5225726</position>
</positions>
...
```

Individual exons can then be visited by piping the record through:

```
xtract -pattern GeneRec -group exons \
  -block positions -pfc "\n" -element position
```

to print a tab-delimited table of start and stop positions:

```
5225463    5225726
5226576    5226799
5226929    5227071
```

Tables to XML

Tab-delimited data is easily converted to XML with xtract **-t2x**:

```
nquire -ftp ftp.ncbi.nlm.nih.gov gene/DATA gene_info.gz |
gunzip -c | grep -v NEWENTRY | cut -f 2,3 |
xtract -t2x -set Set -rec Rec -skip 1 Code Name
```

This takes a series of command-line arguments with tag names for wrapping the individual columns, and skips the first line of input, which contains header information, to generate a new XML file:

```
<Set>
  <Rec>
    <Code>1246500</Code>
    <Name>repA1</Name>
  </Rec>
  <Rec>
    <Code>1246501</Code>
    <Name>repA2</Name>
  </Rec>
  <Rec>
    <Code>1246502</Code>
    <Name>leuA</Name>
  </Rec>
  ...
```

XML Namespaces

Namespace prefixes are indicated by a colon, and a leading colon matches any prefix:

```
nquire -url "http://webservice.wikipathways.org" getPathway -pwId WP455 |
xtract -pattern "ns1:getPathwayResponse" -decode ":gpml" |
```

The **-decode** argument converts Base64-encoded data back to its original binary form. In this case, encoding was used to embed Graphical Pathway Markup Language inside another XML object:

```
xtract -pattern Pathway -block Xref \
  -if @Database -equals "Entrez Gene" \
    -tab "\n" -element @ID
```

Installation

EDirect consists of a set of scripts and programs that are downloaded to the user's computer.

EDirect will run on Unix and Macintosh computers that have the Perl language installed, and under the Cygwin Unix-emulation environment on Windows PCs.

To install the EDirect software, open a terminal window and execute one of the following two commands:

```
sh -c "$(curl -fsSL ftp://ftp.ncbi.nlm.nih.gov/entrez/entrezdirect/install-edirect.sh)"

sh -c "$(wget -q ftp://ftp.ncbi.nlm.nih.gov/entrez/entrezdirect/install-edirect.sh -O -)"
```

or follow the detailed installation instructions in the EDirect web documentation.

This downloads several scripts into an "edirect" folder in the user's home directory. It then fetches any missing Perl modules, and installs platform-specific precompiled executables for xtract and rchive.

At the end of this process, the script will ask for permission to add EDirect to your PATH permanently by editing your configuration file. If you answer "y" it will add:

```
export PATH=${PATH}:$HOME/edirect
```

to the end of your .bash_profile file. If you answer "n", you should then manually edit .bash_profile to add the edirect folder as one of the components of your existing PATH assignment statement.

Documentation

Documentation for EDirect is on the web at:

```
http://www.ncbi.nlm.nih.gov/books/NBK179288
```

EDirect navigation functions call the URL-based Entrez Programming Utilities:

```
https://www.ncbi.nlm.nih.gov/books/NBK25501
```

NCBI database resources are described by:

```
https://www.ncbi.nlm.nih.gov/pubmed/31602479
```

Information on how to obtain an API Key is described in this NCBI blogpost:

```
https://ncbiinsights.ncbi.nlm.nih.gov/2017/11/02/new-api-keys-for-the-e-utilities
```

Additional sample EDirect queries are available from:

```
xtract -examples
```

Questions or comments on EDirect may be sent to **info@ncbi.nlm.nih.gov**.

This research was supported by the Intramural Research Program of the National Library of Medicine at the NIH.