Entrez Direct: E-utilities on the UNIX Command Line

Jonathan Kans, PhD
NCBI
kans@ncbi.nlm.nih.gov
Corresponding author.

Created: April 23, 2013.
Last Update: March 21, 2016.

Getting Started

Introduction

Entrez Direct (EDirect) is an advanced method for accessing the NCBI's set of interconnected databases (publication, sequence, structure, gene, variation, expression, etc.) from a UNIX terminal window. Functions take search terms from command-line arguments. Individual operations are combined to build multi-step queries. Record retrieval and formatting normally complete the process.

EDirect also provides an argument-driven function that simplifies the extraction of data from document summaries or other results that are returned in structured XML format. This can eliminate the need for writing custom software to answer ad hoc questions. Queries can move seamlessly between EDirect commands and UNIX utilities or scripts to perform actions that cannot be accomplished entirely within Entrez.

Installation

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, copy the following commands and paste them into a terminal window:

```bash
cd ~
perl -MNet::FTP -e "
  $ftp->binary; $ftp->get("/entrez/entrezdirect/edirect.zip");"
unzip -u -q edirect.zip
rm edirect.zip
export PATH=$PATH:$HOME/edirect
./edirect/setup.sh
```

This downloads several scripts into an "edirect" folder in the user's home directory, and configures the PATH environment variable to allow execution of programs in that location.

Entrez Direct Functions

EDirect operations can be grouped into several categories.

Navigation functions support exploration within the Entrez databases:

- **esearch** performs a new Entrez search using terms in indexed fields.
• **elink** looks up neighbors (within a database) or links (between databases).
• **efilter** filters or restricts the results of a previous query.

Records can be retrieved in specified formats or as document summaries:
• **efetch** downloads records or reports in a designated format.

Desired fields from XML results can be extracted without writing a program:
• **xtract** converts XML into a table of data values.

Several additional functions are also provided:
• **einfo** obtains information on indexed fields in an Entrez database.
• **epost** uploads unique identifiers (UIDs) or sequence accession numbers.
• **nquire** sends a URL request to a web page or CGI service.

### Entering Query Commands

UNIX programs are run by typing the name of the program and then supplying any required or optional arguments on the command line. Argument names are letters or words that start with a dash ("-".) character.

In order to begin an Entrez search, the user types "esearch" and then enters the required -db (database) and -query arguments. A query on unqualified search terms:

```
esearch -db pubmed -query "opsin gene conversion"
```

constructs the appropriate Entrez Utilities (E-utilities) URL from the query terms and executes the search. EDirect handles many technical details behind the scenes (avoiding the learning curve normally required for E-utilities programming), and saves the results on the Entrez history server.

### Constructing Multi-Step Queries

EDirect gains flexibility by allowing individual operations to be described separately, combining them into a multi-step query by using the vertical bar ("|") UNIX pipe symbol. Piping esearch to elink:

```
esearch -db pubmed -query "opsin gene conversion" | elink -related
```

will look up related articles (precomputed PubMed neighbors) of the initial results.

### Writing Commands on Multiple Lines

A query can be continued on the next line by typing the backslash ("\") UNIX escape character immediately before pressing the Return key. Continuing the query:

```
esearch -db pubmed -query "opsin gene conversion" | \n  elink -related | \n  elink -target protein
```

links to all protein sequences published in the neighbor articles. (The vertical bar pipe symbol at the end of the line also allows the query to continue on the next line.)

### Retrieving PubMed Reports

Piping PubMed query results to efetch and specifying the "abstract" format:
returns a set of reports that can be read by a person:


Levels of lycopene β-cyclase 1 modulate carotenoid gene expression and accumulation in Daucus carota.

Moreno JC(1), Pizarro L, Fuentes P, Handford M, Cifuentes V, Stange C.

Author information:
(1)Departamento de Biología, Facultad de Ciencias, Universidad de Chile, Santiago, Chile.

Plant carotenoids are synthesized and accumulated in plastids through a highly regulated pathway. Lycopene β-cyclase (LCYB) is a key enzyme involved directly in the synthesis of α-carotene and β-carotene through ...

Using "efetch -format medline" instead produces a report that can be entered into common bibliographic management software packages:


Retrieving Sequence Reports

Nucleotide and protein records can be downloaded in FASTA format:

esearch -db protein -query "lycopene cyclase" | efetch -format fasta

which consists of a definition line followed by the sequence:
...gi|735882|gb|AAA81880.1| lycopene cyclase [Arabidopsis thaliana]
MDTLKTPKNDLFIPQFHGFERLCSNNPYESRVRGLGKRAIKIVSSVSAGALLDLPETKKEMLDF
ELPLYDTKSQVVDVLA1VVGGPGAGLAQYQVEAGLSCISDPSKLNWNYPNYGVWDEFEAMDLDDLC
DDTDGGAVVYDEGKDLRPFYQRNVRKQLSKMLQKCI1NGVQFKQSTVNVHEANSTVCCSGDVKQ
QASVVLATGFSLVQYDKPYNPGYQVAYGILIAEVDGHFPFDVDMVFMDWRDKHLSYPELKERNSKIP
TLHMPFSSNRWFLEETSLLFVRFLGAMEDIQERMAARLKLNLGIVNVEEEDERCVIPMGFLFPVLQF
VQ1GTAGMHPSTQYVXARLAAAIATANVIVRNYLGSPPSSNLSKPGDQLSAEVRDLWPIERRRRQREF
FCFGMDILKLDLDAITRFFDAFFDLQPYHWHFLSSRRLFEPULLVFLGSLFHSASNTSRLEIMTGTVP
LAKMNNLVQDRD

Piping to a UNIX "grep" command removes blank lines between FASTA records:

grep '.'

Sequence records can also be obtained in GenBank (-format gb) or GenPept (-format gp)
flatfile formats, which have features annotating particular regions of the sequence:

...LOCUS AAA81880 501 aa linear PLN ...
DEFINITION lycopene cyclase [Arabidopsis thaliana].
ACCESSION AAA81880
VERSION AAA81880.1 GI:735882
DBSOURCE locus ATLYC accession L40176.1
KEYWORDS .
SOURCE Arabidopsis thaliana (thale cress)
ORGANISM Arabidopsis thaliana
  Eukaryota; Viridiplantae; Streptophyta; Embryophyta;
  Tracheophyta; Spermatophyta; Magnoliophyta; eudicotyledons;
  Brassicales; Brassicaceae; Camelineae; Arabidopsis.
REFERENCE 1 (residues 1 to 501)
AUTHORS Scolnik,P.A. and Bartley,G.E.
TITLE Nucleotide sequence of lycopene cyclase (GenBank L40176) from
Arabidopsis (PGR95-019)
...
FEATURES Location/Qualifiers
  source 1..501
    /organism="Arabidopsis thaliana"
    /db_xref="taxon:3702"
  Protein 1..501
    /product="lycopene cyclase"
  transit_peptide 1..80
  mat_peptide 81..501
    /product="lycopene cyclase"
  CDS 1..501
    /gene="LYC"
    / coded_by="L40176:1:2..1507"
ORIGIN
 1 mdtllktpnk ldffipqfhg ferlcsnnpyp sarvrlgvk raikivsvvv szgaallldiv
 61 petkkenldf elplytdaks qvvdlaivgg gpaglavaqq vseaglsvcs idpskliwp

Entrez Direct: E-utilities on the UNIX Command Line
These formats are suitable for input into many popular sequence analysis programs.

**Searching and Filtering**

**Restricting Query Results**

The current results can be refined by further term searching in Entrez (useful in the protein database for limiting BLAST neighbors to a taxonomic subset):

```
esearch -db pubmed -query "opsin gene conversion" |
elink -related |
efilter -query "tetrachromacy"
```

Results can also be filtered by time. For example, the following statements:

```
efilter -days 60 -datetype PDAT
efilter -mindate 1990 -maxdate 1999 -datetype PDAT
```

restrict results to articles published in the previous two months or in the 1990s, respectively.

**Qualifying Queries by Indexed Field**

Query terms in esearch or efilter can be qualified by entering an indexed field abbreviation in brackets. Boolean operators and parentheses can also be used in the query expression for more complex searches.

Commonly-used fields for PubMed queries include:

- [AFFL] Affiliation
- [FILT] Filter
- [MESH] MeSH Terms
- [ALL] All Fields
- [JOUR] Journal
- [PTYP] Publication Type
- [AUTH] Author
- [LANG] Language
- [WORD] Text Word
- [FAUT] Author - First
- [MAJR] MeSH Major Topic
- [TITL] Title
- [LAUT] Author - Last
- [SUBH] MeSH Subheading
- [TIAB] Title/Abstract
- [PDAT] Date - Publication
- [UID] UID

and a qualified query looks like:

"Tager HS [AUTH] AND glucagon [TIAB]"

Filters that limit search results to subsets of PubMed include:

- humans [MESH] has abstract [FILT]
- pharmacokinetics [MESH] historical article [FILT]
- chemically induced [SUBH] loprovflybase [FILT]
Sequence databases are indexed with a different set of search fields, including:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCN</td>
<td>Accession</td>
</tr>
<tr>
<td>GENE</td>
<td>Gene Name</td>
</tr>
<tr>
<td>PROT</td>
<td>Protein Name</td>
</tr>
<tr>
<td>ALL</td>
<td>All Fields</td>
</tr>
<tr>
<td>JOUR</td>
<td>Journal</td>
</tr>
<tr>
<td>SQID</td>
<td>SeqID String</td>
</tr>
<tr>
<td>AUTH</td>
<td>Author</td>
</tr>
<tr>
<td>KYWD</td>
<td>Keyword</td>
</tr>
<tr>
<td>MLWT</td>
<td>Molecular Weight</td>
</tr>
<tr>
<td>SUBS</td>
<td>Substance Name</td>
</tr>
<tr>
<td>GPRJ</td>
<td>BioProject</td>
</tr>
<tr>
<td>ECNO</td>
<td>EC/RN Number</td>
</tr>
<tr>
<td>ORGN</td>
<td>Organism</td>
</tr>
<tr>
<td>WORD</td>
<td>Text Word</td>
</tr>
<tr>
<td>FKEY</td>
<td>Feature Key</td>
</tr>
<tr>
<td>PACC</td>
<td>Primary Accession</td>
</tr>
<tr>
<td>FILT</td>
<td>Filter</td>
</tr>
<tr>
<td>PROP</td>
<td>Properties</td>
</tr>
<tr>
<td>UID</td>
<td>UID</td>
</tr>
</tbody>
</table>

and a sample query in the protein database is:

"alcohol dehydrogenase [PROT] NOT (bacteria [ORGN] OR fungi [ORGN])"

Additional examples of subset filters in sequence databases are:

- mammalia [ORGN]
- dbxref flybase [PROP]
- mammalia [ORGN:noexp]
- gbdiv phg [PROP]
- cds [FKEY]
- phylogenetic study [PROP]
- lacz [GENE]
- sequence from mitochondrion [PROP]
- beta galactosidase [PROT]
- src cultivar [PROP]
- protein snp [FILT]
- srcdb refseq validated [PROP]
- reviewed [FILT]
- 150:200 [SLEN]
- biomol genomic [PROP]
- 2000:4000 [MLWT]

(The calculated molecular weight (MLWT) field is only indexed for proteins (and structures), not nucleotides.)

**Examining Intermediate Results**

EDirect stores intermediate results on the Entrez history server. EDirect navigation functions produce a custom XML message with the relevant fields (database, web environment, query key, and record count) that can be read the next command in the pipeline.

The results of each step in a query can be examined to confirm expected behavior before adding the next step. The Count field in the ENTREZ_DIRECT object contains the number of records returned by the previous step. A good measure of query success is a reasonable (non-zero) count value. For example:

```
esearch -db protein -query "NP_567004 [ACCN]" |
elink -related |
efilter -query "28000:30000 [MLWT]" |
elink -target structure |
efilter -query "0:2 [RESO]"
```

produces:

```
<ENTREZ_DIRECT>
<Db>structure</Db>
```
with 39 protein structures being within the specified molecular weight range and having the
desired (X-ray crystallographic) atomic position resolution.

(The QueryKey value is 7 instead of 5 because each elink command obtains the record count
by running a separate ESearch query immediately after the ELink operation.)

**Combining Independent Queries**

Independent esearch, elink, and efilter operations can be performed and then combined at
the end by using the history server’s “#” convention to indicate query key numbers. (The
steps to be combined must be in the same database.) Subsequent esearch commands can take
a -db argument to override the database piped in from the previous step. (Piping the queries
together is necessary for sharing the same history thread.) For example, the query:

```
esearch -db protein -query "amyloid* [PROT]" |
elink -target pubmed |
esearch -db gene -query "apo* [GENE]" |
elink -target pubmed |
esearch -query "(#3) AND (#6)" |
efetch -format docsum |
xtract -pattern DocumentSummary -element Id Title
```

uses truncation searching (entering the beginning of a word followed by an asterisk) to
return titles of papers with links to amyloid protein sequence and apolipoprotein gene
records:

```
23962925    Genome analysis reveals insights into physiology and ...
23959870    Low levels of copper disrupt brain amyloid-β homeostasis ...
23371554    Genomic diversity and evolution of the head crest in the ...
23251661    Novel genetic loci identified for the pathophysiology of ...
...
```

The use of (#3) AND (#6) instead of (#2) AND (#4) above reflects the need for each elink
command to execute a separate ESearch query, which increments the QueryKey, in order
to obtain the record count. The -label argument can be used to get around this artifact. The
label value is prefixed by a “#” symbol and placed in parentheses in the final search. Thus:

```
esearch -db structure -query "insulin [TITL]" |
elink -target pubmed -label struc_cit |
esearch -db protein -query "insulin [PROT]" |
elink -target pubmed -label prot_cit |
esearch -query "(#struc_cit) AND (#prot_cit)" |
efetch -format uid
```

will return:
without the need to keep track of the internal QueryKey values.

**Structured Data**

**Advantages of XML Format**

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

The advantage of XML is that many pieces of information are in specific locations in a well-defined data hierarchy. Accessing individual units of data that are fielded by name, such as:

```
<PubDate>2013</PubDate>
<Source>PLoS One</Source>
<Volume>8</Volume>
<Issue>3</Issue>
<Pages>e58144</Pages>
```

requires matching the same general pattern, differing only by the element name. This is much simpler than parsing the units from a long, complex string:

```
```

The disadvantage of XML is that data extraction usually requires programming. But EDirect relies on the common pattern of XML value representation to provide a simplified approach to interpreting XML data.

**Conversion of XML Data into Tabular Form**

The xtract function uses command-line arguments to direct the selective conversion of XML data into a tab-delimited table. The -pattern argument divides the results into rows, while placement of data into columns is controlled by -element. A trivial example:

```
xtract -pattern ENTREZ_DIRECT -element Count
```

will print the number of records in the current query.

Xtract provides control over data conversion with a divide-and-conquer strategy using separate arguments for element selection, path exploration, conditional processing, and report formatting.

Element selection finds every occurrence of each indicated item, printing values as they are encountered. Exploration control limits selection by context, presenting specified objects one at a time. Conditional processing filters by content, requiring presence (or absence) of a particular data value in order to continue. Finally, custom formatting can override the normal tabular layout of the default output.
The details and ramifications of this flexible approach are discussed in the remainder of this section.

**Extraction Arguments**

Selection arguments (-element, -first, and -last) extract and print data values from the indicated element names:

```-element Id -first Name Title```

Exploration arguments (-pattern, -group, -block, and -subset) limit data extraction to specified regions of the XML, visiting all relevant objects one at a time. This sets a context for data collection, eliminates the need to provide the full path to a data element, and uncouples the concept of "what to look for" from "where to find it":

```-pattern DocumentSummary
   -block Author```

Each pattern can have multiple groups, each group can have multiple blocks, and each block can have multiple subsets. This design allows nested exploration of complex, hierarchical data to be controlled by a linear chain of command-line argument statements.

Conditional processing arguments restrict exploration statements by object name (-match and -avoid) or item location (-position):

```-match "Source:J Bacteriol"
   -position first```

These commands are issued immediately after an exploration argument.

(The -match and -avoid arguments can use an "Element:Value" construct to specify an element with a particular value.)

Formatting arguments (-ret, -tab, -sep, -pfx, and -sfx) allow extensive customization of the default row/column table presentation:

```-pfx "\n[" -sfx "]\t" -sep " " -tab "" -ret "\n\n"
```

and apply to subsequent selection statements.

(The "\n" escape sequence indicates a line break, while "\t" specifies a tab character.)

**XML Document Summaries**

Entrez provides a document summary in structured XML format for every record. Piping a query to "efetch -format docsum":

```esearch -db pubmed -query "Garber ED [AUTH] AND PNAS [JOUR]" | elink -related | efilter -query "mouse" | efetch -format docsum```

will generate an XML document summary set:
Piping the document summary output to:

\texttt{xtract -outline}

will give an indented overview of the XML structure hierarchy:

\begin{verbatim}
DbBuild
DocumentSummary
 Id
 PubDate
 EPubDate
 Source
 Authors
 Author
 Name
 AuthType
 ClusterID
 Author
 Name
 ... 
\end{verbatim}

The outline view presents a clear, uncluttered picture of the XML hierarchy that is useful in designing the appropriate command for actual data extraction. Copy and paste from the -outline output to xtract arguments can help avoid typographical errors. Thus:

\begin{verbatim}
esearch -db pubmed -query "Garber ED [AUTH] AND PNAS [JOUR]" |
elink -related |
efilter -query "mouse" |
efetch -format docsum |
xtract -pattern DocumentSummary -element Id SortFirstAuthor Title
\end{verbatim}

returns the PubMed identifier (PMID), first author name, and article title:
Processing Results with UNIX Utilities

A tab-delimited table can be processed by many UNIX utilities. For example:

```
esearch -db pubmed -query "Garber ED [AUTH] AND PNAS [JOUR]" |
elink -related |
efilter -query "mouse" |
efetch -format docsum |
xtract -pattern DocumentSummary -element Id SortFirstAuthor Title |
sort -t $'	' -k 2,2f -k 3,3f
```

sorts the results of the previous example by author name and then (if there are multiple publications by the same author) alphabetically by title:

```
17474906    Benghezal M     Inhibitors of bacterial virulence ...
19650888    Cano V          Klebsiella pneumoniae triggers a cytotoxic ...
17102561    Chatterjee S    How reliable are models for malaria vaccine ...
17371870    Clements A      Secondary acylation of Klebsiella ...
17142396    Fresno S        A second galacturonic acid transferase is ...
16735743    Fresno S        The ionic interaction of Klebsiella ...
19248821    Fukumoto N      Hypoalgesic behaviors of P/Q-type voltage- ...
... 
```

Rather than always having to retype a series of common post-processing instructions, frequently used combinations of UNIX commands can be placed in a function, stored in an alias file (e.g., the user's .bash_profile), and executed by name. (The following two functions are now included as scripts with the EDirect software.) For example:

```
WordAtATime() {
    sed 's/[^a-zA-Z0-9]/ /g; s/^ */://' |
    tr 'A-Z' 'a-z' |
    fmt -w 1 |
}
alias word-at-a-time='WordAtATime'

SortUniqCountRank() {
    sort -f |
    uniq -i -c |
    perl -e 's/$/s*$\d*$/s{1}<s\d*>/s/' |
    sort -t $'	' -k 1,1nr -k 2f |
}
alias sort-uniq-count-rank='SortUniqCountRank'
```

Titles can be passed to a pair of these UNIX alias commands:
esearch -db pubmed -query "Casadaban transposition immunity" |
elink -related |
efetch -format docsum |
xtract -pattern DocumentSummary -element Title |
word-at-a-time |
sort-uniq-count-rank

to generate a table of word occurrence counts, sorted by frequency:

<table>
<thead>
<tr>
<th>Count</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>299</td>
<td>of</td>
</tr>
<tr>
<td>178</td>
<td>the</td>
</tr>
<tr>
<td>114</td>
<td>transposition</td>
</tr>
<tr>
<td>100</td>
<td>and</td>
</tr>
<tr>
<td>94</td>
<td>in</td>
</tr>
<tr>
<td>93</td>
<td>mu</td>
</tr>
<tr>
<td>82</td>
<td>a</td>
</tr>
<tr>
<td>61</td>
<td>dna</td>
</tr>
<tr>
<td>61</td>
<td>tn3</td>
</tr>
<tr>
<td>56</td>
<td>transposon</td>
</tr>
<tr>
<td>50</td>
<td>bacteriophage</td>
</tr>
</tbody>
</table>

**Output Format Customization**

The line break between -pattern objects can be overridden with -ret, and the tab character between fields can be replaced by -tab.

The -sep argument is used to distinguish multiple elements of the same type and control their separation independently of the -tab argument. For example:

esearch -db gene -query "deuteranopia" |
efetch -format xml |
xtract -pattern Entrezgene \ 
    -element Gene-track_geneid Gene-ref_locus \ 
    -sep "|" -element Gene-ref_syn_E

combines all synonyms for a gene into a single column, separated by vertical bars:

<table>
<thead>
<tr>
<th>Gene-ID</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>2652</td>
<td>OPN1MW</td>
</tr>
<tr>
<td>5956</td>
<td>OPN1LW</td>
</tr>
</tbody>
</table>

The -sep value also applies to unrelated -element items that are grouped with commas. Otherwise the -tab value delineates individual fields.

Groups or isolated fields are preceded by the -pfx value and followed by the -sfx value, both of which are initially empty.

**Pubmed Article XML Records**

The PubmedArticle object has a more detailed structure than the DocumentSummary, and is available for records in the pubmed database:
esearch -db pubmed -query "tetrachromacy" |
efetch -format xml |
xtract -outline

More information is fielded, including author names, dates, and the abstract:

PubmedArticle
  MedlineCitation
    PMID
    DateCreated
      Year
      Month
      Day
    DateCompleted
      Year
      Month
      Day
    DateRevised
      Year
      Month
      Day
  Article
    Journal
      ISSN
    JournalIssue
      Volume
      Issue
      PubDate
        Year
        Month
        Day
    Title
    ISOAbbreviation
  ArticleTitle
  Pagination
    MedlinePgn
    ELocationID
  Abstract
    AbstractText
  CopyrightInformation
  AuthorList
    Author
      LastName
      ForeName
      Initials
      AffiliationInfo
        Affiliation
    Author
      LastName
      ...

Using this information to craft a new xtract statement:

Entrez Direct: E-utils on the UNIX Command Line
esearch -db pubmed -query "tetrachromacy" |
efetch -format xml |
xtract -pattern PubmedArticle -element MedlineCitation/PMID LastName

results in a table of all authors for each record:

<table>
<thead>
<tr>
<th>PMID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Initials</th>
<th>Last Name</th>
<th>First Name</th>
<th>Initials</th>
</tr>
</thead>
<tbody>
<tr>
<td>23393278</td>
<td>Sabbah</td>
<td>Troje</td>
<td>Gray</td>
<td>Hawryshyn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20884587</td>
<td>Jordan</td>
<td>Deeb</td>
<td>Bosten</td>
<td>Mollon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18230593</td>
<td>Koshitaka</td>
<td>Kinoshita</td>
<td>Vorobyev</td>
<td>Arikawa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17685813</td>
<td>Wachtler</td>
<td>Doi</td>
<td>Lee</td>
<td>Sejnowski</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16086150</td>
<td>Goldsmith</td>
<td>Butler</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...  

(Note that "-element MedlineCitation/PMID" uses the "Parent/Child" construct to prevent the display of additional PMID items that may occur later in CommentsCorrections objects.)

The -first or -last arguments can be used instead of -element, if appropriate.

**Exploration of XML Sets**

Individual PubmedArticle objects can be retrieved directly by efetch:

```
efetch -db pubmed -id 20643751 -format xml
```

The resulting XML has authors with separate fields for last name and initials:

```
...<AuthorList>
  <Author>
    <LastName>Inamdar</LastName>
    <ForeName>Arati A</ForeName>
    <Initials>AA</Initials>
  </Author>
  <Author>
    <LastName>Masurekar</LastName>
    <ForeName>Prakash</ForeName>
    <Initials>P</Initials>
  </Author>
  <Author>
    <LastName>Bennett</LastName>
    <ForeName>Joan Wennstrom</ForeName>
    <Initials>JW</Initials>
  </Author>
</AuthorList>
...
```

Without being given any guidance about context, an -element statement with "Initials" and "LastName" arguments:

```
efetch -db pubmed -id 1413997,6301692,781293 -format xml |
xtract -pattern PubmedArticle -element MedlineCitation/PMID -element Initials LastName
```
will print all author initials and then all author last names:

1413997    RK    CR           JS         Mortimer      Contopoulou    King
6301692    MA    NR           Krasnow    Cozzarelli
781293     MJ    Casadaban

A -block statement redirects data exploration to visit each author one at a time. Subsequent -element statements only see the current object's values:

```
efetch -db pubmed -id 1413997,6301692,781293 -format xml |
xtract -pattern PubmedArticle -element MedlineCitation/PMID 
   -block Author -element Initials LastName
```

which restores the correct association of initials and last name:

1413997    RK    Mortimer     CR    Contopoulou    JS    King
6301692    MA    Krasnow      NR    Cozzarelli
781293     MJ    Casadaban

Adding a -sep statement to replace the normal tab between group members, and using a comma to combine the two arguments ("Initials,LastName") into a group:

```
efetch -db pubmed -id 1413997,6301692,781293 -format xml |
xtract -pattern PubmedArticle -element MedlineCitation/PMID 
   -block Author -sep " " -element Initials,LastName
```

results in the proper pairing of author field values along with more desirable formatting:

1413997    RK Mortimer     CR Contopoulou    JS King
6301692    MA Krasnow      NR Cozzarelli
781293     MJ Casadaban

**Recording Values in Variables**

A value can be recorded in a variable and then displayed multiple times as needed. Variables are indicated by a hyphen followed by a string of capital letters or digits. The variable "&PMID" is referred to as "&PMID" in an -element argument. For example:

```
efetch -db pubmed -id 1413997,6301692,781293 -format xml |
xtract -pattern PubmedArticle -element MedlineCitation/PMID 
   -block Author -element "&PMID" 
   -sep " " -tab "\n" -element Initials,LastName
```

produces a list of authors, with the PMID in the first column of each row:

1413997    RK Mortimer
1413997    CR Contopoulou
1413997    JS King
6301692    MA Krasnow
6301692    NR Cozzarelli
781293     MJ Casadaban

Variables can be initialized with a literal value in parentheses:
This can be used as a placeholder to prevent missing data from shifting columns in a table, or to have additional control over output formatting:

1413997    RK Mortimer, CR Contopoulou, JS King
6301692    MA Krasnow, NR Cozzarelli
781293     MJ Casadaban

Exploring Separate XML Regions

Multiple -block statements can be used in a single xtract to explore different areas of the XML. This limits element extraction to the desired subregions, and allows disambiguation of fields with identical names.

Combining independent fields with commas allows them to be treated as sets. The tab that normally separates these can be replaced with a -sep argument:

This generates a table that allows easy parsing of author last names, counts the number of authors present, and prints the date each record was published and the date it was entered into PubMed, sorting the results by the computed author count:

( withheld for brevity)
Nested Exploration of Subsets Within XML Sets

Medical Subject Headings (MeSH terms) in a record may be assigned subheadings:

\[
\begin{align*}
\text{<MeshHeading>}
\text{<DescriptorName>RNA, Messenger</DescriptorName>}
\text{<QualifierName>genetics</QualifierName>}
\text{</MeshHeading>}
\text{<MeshHeading>}
\text{<DescriptorName>Transcription, Genetic</DescriptorName>}
\text{</MeshHeading>}
\text{<MeshHeading>}
\text{<DescriptorName>beta-Galactosidase</DescriptorName>}
\text{<QualifierName>genetics</QualifierName>}
\text{<QualifierName>metabolism</QualifierName>}
\text{</MeshHeading>}
\text{</MeshHeadingList>}
\end{align*}
\]

Visiting each MeSH term with a -block statement, and adding a -subset statement within the -block, allows nested exploration of the subheadings for the current MeSH term:

```
efetch -db pubmed -id 6162838 -format xml |
xtract -pattern PubmedArticle -tab " " -element MedlineCitation/PMID \
   -block MeshHeading -pfx "\n" -tab " " -element DescriptorName \
   -subset QualifierName -pfx "/" -tab " " -element QualifierName
```

and creates a list of MeSH terms with associated subheadings:

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

The MeSH term and subheading fields actually have major topic attributes:

```
\begin{align*}
\text{<MeshHeading>}
\text{<DescriptorName MajorTopicYN="N">beta-Galactosidase</DescriptorName>}
\text{<QualifierName MajorTopicYN="Y">genetics</QualifierName>}
\text{<QualifierName MajorTopicYN="N">metabolism</QualifierName>}
\text{</MeshHeading>}
\end{align*}
```

that can be selected as "DescriptorName@MajorTopicYN" or "@MajorTopicYN":

Entrez Direct: E-utilities on the UNIX Command Line
The major topic value is placed before each MeSH term or subheading:

6162838
|N|Base Sequence
|Y|DNA, Recombinant
|N|Escherichia coli/|N|genetics
...
|N|RNA, Messenger/|Y|genetics
|N|Transcription, Genetic
|N|beta-Galactosidase/|Y|genetics/|N|metabolism

The results can be processed by the UNIX stream editor "sed":

```
sed -e 's/|N|//g' -e 's/|Y|/*/g'
```

to display an asterisk for major ("starred" MeSH term) concepts:

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

**Conditional Processing**

Xtract provides -match and -avoid arguments that filter by element name or name plus data value. Parallel statements are used to handle alternative conditions. For example:

```
efetch -db pubmed -id 6162838 -format xml |
xtract -pattern PubmedArticle -PMID MedlineCitation/PMID \ 
 -group MeshHeading \ 
  -block MeshHeading -match QualifierName \ 
   -subset DescriptorName -TERM "DescriptorName" -MAJR "@MajorTopicYN" \ 
   -subset QualifierName -tab "\n" \ 
   -element "&PMID","&TERM","&MAJR",QualifierName,"@MajorTopicYN" \ 
  -block MeshHeading -avoid QualifierName \ 
   -subset DescriptorName -tab "\n" \ 
   -element "&PMID",DescriptorName,"@MajorTopicYN"
```

has separate sections for MeSH terms with and without subheadings, and produces results that are suitable for importing into a database or spreadsheet program:
Multiple `-match` or `-avoid` conditions are specified with `-and` and `-or` commands.

**Sequence Records**

**NCBI Data Model for Sequence Records**

The NCBI represents sequence records in a data model that is based on the central dogma of molecular biology. Sequences, including genomic DNA, messenger RNAs, and protein products, are “instantiated” with the actual sequence letters, and are assigned identifiers (e.g., accession numbers) for reference. Features carry information about the biology of a given region, with a location that refers to specific intervals on a particular sequence. Some features may also point to the product sequence of a particular transformation.
A gene feature indicates the location of a heritable region of nucleic acid that confers a measurable phenotype. An mRNA feature on genomic DNA represents the exonic and untranslated regions of the message that remain after transcription and splicing. A coding region (CDS) feature has a product reference to the translated protein.

Since messenger RNA sequences are not always submitted with a genomic region, CDS features (which model the travel of ribosomes on transcript molecules) are traditionally annotated on the genomic sequence, with locations that encode the exonic intervals.

Features display specific biological annotation in qualifiers. For example, the name of a gene is shown in the /gene qualifier. A qualifier can be dynamically generated from underlying data for the convenience of the user. Thus, the sequence of a mature peptide may be extracted from the mat_peptide feature’s location on the precursor protein and displayed in a /peptide qualifier, even if a mature peptide is not instantiated.
Sequence Records in INSDSeq XML

Sequence records can be retrieved in an XML version of the GenBank or GenPept flatfile. The query:

```
efetch -db protein -id 26418308,26418074 -format gpc
```

returns a set of INSDSeq objects:

```
<INSDSet>
  <INSDSeq>
    <INSDSeq_locus>AAN78128</INSDSeq_locus>
    <INSDSeq_length>17</INSDSeq_length>
    <INSDSeq_moltype>AA</INSDSeq_moltype>
    <INSDSeq_topology>linear</INSDSeq_topology>
    <INSDSeq_division>INV</INSDSeq_division>
    <INSDSeq_update-date>03-JAN-2003</INSDSeq_update-date>
    <INSDSeq_create-date>10-DEC-2002</INSDSeq_create-date>
    <INSDSeq_definition>alpha-conotoxin ImI precursor, partial [Conus imperialis]</INSDSeq_definition>
    <INSDSeq_primary-accession>AAN78128</INSDSeq_primary-accession>
    <INSDSeq_accession-version>AAN78128.1</INSDSeq_accession-version>
    <INSDSeq_other-seqids>
      <INSDSeqid>gb|AAN78128.1|</INSDSeqid>
      <INSDSeqid>gi|26418308</INSDSeqid>
    </INSDSeq_other-seqids>
    <INSDSeq_source>Conus imperialis</INSDSeq_source>
    <INSDSeq_organism>Conus imperialis</INSDSeq_organism>
    <INSDSeq_taxonomy>Eukaryota; Metazoa; Lophotrochozoa; Mollusca; Gastropoda; Caenogastropoda; Hypsogastropoda; Neogastropoda; Conoidea; Conidae; Conus</INSDSeq_taxonomy>
    <INSDSeq_references>
      ...
    </INSDSeq_references>
  </INSDSeq>
</INSDSet>
```

INSDSeq XML presents biological features and qualifiers (shown here in GenPept format):

<table>
<thead>
<tr>
<th>FEATURES</th>
<th>Location/Qualifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>1..17</td>
</tr>
<tr>
<td></td>
<td>/organism=&quot;Conus imperialis&quot;</td>
</tr>
<tr>
<td></td>
<td>/db_xref=&quot;taxon:35631&quot;</td>
</tr>
<tr>
<td></td>
<td>/country=&quot;Philippines&quot;</td>
</tr>
<tr>
<td>Protein</td>
<td>&lt;1..17</td>
</tr>
<tr>
<td></td>
<td>/product=&quot;alpha-conotoxin ImI precursor&quot;</td>
</tr>
<tr>
<td>mat_peptide</td>
<td>5..16</td>
</tr>
<tr>
<td></td>
<td>/product=&quot;alpha-conotoxin ImI&quot;</td>
</tr>
<tr>
<td></td>
<td>/note=&quot;the C-terminal glycine of the precursor is post translationally removed&quot;</td>
</tr>
<tr>
<td></td>
<td>/calculated_mol_wt=1357</td>
</tr>
<tr>
<td></td>
<td>/peptide=&quot;GCCSDPRCAWRC&quot;</td>
</tr>
<tr>
<td>CDS</td>
<td>1..17</td>
</tr>
<tr>
<td></td>
<td>/coded_by=&quot;AY159318.1:&lt;1..54&quot;</td>
</tr>
<tr>
<td></td>
<td>/note=&quot;nAChR antagonist&quot;</td>
</tr>
</tbody>
</table>

Entrez Direct: E-utils on the UNIX Command Line
in a structured feature table:

...  
<INSDFeature>
  <INSDFeature_key>mat_peptide</INSDFeature_key>
  <INSDFeature_location>5..16</INSDFeature_location>
  <INSDFeature_intervals>
    <INSDInterval>
      <INSDInterval_from>5</INSDInterval_from>
      <INSDInterval_to>16</INSDInterval_to>
      <INSDInterval_accession>AAN78128.1</INSDInterval_accession>
    </INSDInterval>
  </INSDFeature_intervals>
  <INSDFeature_quals>
    <INSDQualifier>
      <INSDQualifier_name>product</INSDQualifier_name>
      <INSDQualifier_value>alpha-conotoxin ImI</INSDQualifier_value>
    </INSDQualifier>
    <INSDQualifier>
      <INSDQualifier_name>note</INSDQualifier_name>
      <INSDQualifier_value>the C-terminal glycine of the precursor is post translationally removed</INSDQualifier_value>
    </INSDQualifier>
    <INSDQualifier>
      <INSDQualifier_name>calculated_mol_wt</INSDQualifier_name>
      <INSDQualifier_value>1357</INSDQualifier_value>
    </INSDQualifier>
    <INSDQualifier>
      <INSDQualifier_name>peptide</INSDQualifier_name>
      <INSDQualifier_value>GCCSDPRCAWRC</INSDQualifier_value>
    </INSDQualifier>
  </INSDFeature_quals>
</INSDFeature>

Feature and qualifier names are indicated in data values, not XML element tags, and require -match to select the desired object and content. The xtract -insd argument simplifies this process, as shown below.

**Generating Qualifier Extraction Commands**

Because obtaining specific qualifier values from INSDSeq XML is somewhat more complex than previous cases, the xtract -insd argument can be used to generate extraction instructions.

Running xtract -insd in an isolated command prints a new xtract statement that can then be copied, edited if necessary, and pasted into other queries. Running the -insd command within a multi-step pipe dynamically executes the constructed query.

Providing an optional (complete/partial) location indication, a feature key, and then one or more qualifier names:

```
xtract -insd complete mat_peptide "%peptide" product peptide
```
creates a new xtract statement that will produce a table of qualifier values from mature
peptide features with complete locations. The statement starts with instructions to record the
accession and find features of the indicated type:

```bash
xtract -pattern INSDSeq -ACCN INSDSeq_accession-version \ 
  -group INSDFeature -match INSDFeature_key:mat_peptide \ 
  -avoid INSDFeature_partial5 -and INSDFeature_partial3 \ 
  -pfx "\n" -element "$ACCN"
```

Each qualifier argument will then generate custom extraction code that is appended to the
growing query. For peptide this produces:

```bash
-block INSDQualifier \ 
  -match INSDQualifier_name:peptide \ 
  -element INSDQualifier_value
```

Incorporating the xtract -insd command in a query for marine snail venom peptides:

```bash
esearch -db pubmed -query "conotoxin" | 
elink -target protein | 
efilter -query "mat_peptide [FKEY]" | 
efetch -format gpc | 
xtract -insd complete mat_peptide "$peptide" product peptide
```

produces a table with columns for accession number, calculated peptide length, product
name, and peptide sequence:

<table>
<thead>
<tr>
<th>Accession</th>
<th>Length</th>
<th>Product Name</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGO59814.1</td>
<td>32</td>
<td>del13b conotoxin</td>
<td>DCPTSCPPTTCANGWECCKGYPCVRQHCSGCNH</td>
</tr>
<tr>
<td>AAO33169.1</td>
<td>16</td>
<td>alpha-conotoxin GIC</td>
<td>GCCSHPACAGNNHIC</td>
</tr>
<tr>
<td>ADB65788.1</td>
<td>20</td>
<td>conotoxin Cal 16</td>
<td>LEMQGCVNANAKFCGGEGR</td>
</tr>
<tr>
<td>AAN78128.1</td>
<td>12</td>
<td>alpha-conotoxin ImI</td>
<td>GCCSDPRCAWRC</td>
</tr>
<tr>
<td>AAF23167.1</td>
<td>31</td>
<td>BeTX toxin</td>
<td>CRAEGTYCENDSQCCLNECCWGCGRPCRHP</td>
</tr>
<tr>
<td>ADB65789.1</td>
<td>20</td>
<td>conotoxin Cal 16</td>
<td>LEMQGCVNANAKFCGGEGR</td>
</tr>
<tr>
<td>AAN78279.1</td>
<td>21</td>
<td>conotoxin Vx-II</td>
<td>WIDPSHYCCGCGTDCCVNC</td>
</tr>
<tr>
<td>ABW16858.1</td>
<td>15</td>
<td>marmophin</td>
<td>DWEYHAAHPKPSFWT</td>
</tr>
</tbody>
</table>

Piping the results to a series of UNIX commands:

```bash
grep -i conotoxin | 
awk -F ' \t' -v 'OFS=\t' '{if ( 10 <= $2 && $2 <= 30 ) print}' | 
sort -t $'\t' -u -k 3,4 | 
sort -t $'\t' -k 2,2n -k 3,3f | 
cut -f 1,3- | 
column -s $'\n' -t
```

filters by product name, limits the results to a specified range of peptide lengths, removes
redundant accessions, sorts the table by peptide length, deletes the length column, and aligns
the columns for cleaner printing:

<table>
<thead>
<tr>
<th>Accession</th>
<th>Product Name</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAN78128.1</td>
<td>alpha-conotoxin ImI</td>
<td>GCCSDPRCAWRC</td>
</tr>
<tr>
<td>AAN78279.1</td>
<td>alpha-conotoxin ImII</td>
<td>ACCSDDRRCRWRCA</td>
</tr>
</tbody>
</table>

*Entrez Direct: E-utilities on the UNIX Command Line*
Extensive Modification of Tabular Output

The normal column-oriented output can be modified to produce a custom report. A query on a gene that undergoes messenger RNA splicing:

```
efetch -db nucore -id "GQ370762.1" -format gbc |
| xtract -pattern INSDSeq |
| -pfx "">Feature" -tab "" -first INSDSeqid |
| -group INSDFeature |
| -avoid "INSDFeature_key:source" |
| -FKEY INSDFeature_key |
| -block INSDInterval |
| -pfx "\n" -tab "" |
| -element INSDInterval_from,INSDInterval_to |
| INSDInterval_point,INSDInterval_point |
| -pfx "\t" -tab "" |
| -element "&FKEY" -FKEY "()" |
| -block INSDQualifier |
| -avoid "INSDQualifier_name:transcription" |
| -and "INSDQualifier_name:translation" |
| -and "INSDQualifier_name:peptide" |
| -pfx "\n|\t|\t" -tab "" |
| -element INSDQualifier_name,INSDQualifier_value
```
clears the feature key variable after its first use to reproduce the 5-column feature table format used for GenBank sequence record submissions:

```
>Feature gb|GQ370762.1|
51   1474     gene
    gene   HBB
51    142    mRNA
273   495
1346  1474
```
A location interval will have either an INSDInterval_from and INSDInterval_to pair or a single INSDInterval_point. Additional work would be required to support the 5' and 3' partial flags for features with incomplete locations.

The actual 5-column feature table representation of any sequence record can be obtained directly by using "efetch -format ft".

**Advanced Topics**

**Storing Common Phrases in Alias Files**

Long or complicated search phrases can be saved in a file to avoid having to retype (or copy and paste) the full text for each query. Each line of the file has a shortcut keyword, a tab character, and the expanded search term. Shortcuts are referenced by placing them in parentheses after prefixing with a pound ("#") sign.

For example, given a file named "q_aliases" containing:

```plaintext
jour_filt   [MULT] AND ncbijournals [FILT]
trans_imm   (transposition OR target) immunity
```

the esearch line in:

```plaintext
esearch -alias q_aliases -db nlmcatalog -query "Science (#jour_filt)"
```

will be expanded to:

```plaintext
esearch -db nlmcatalog -query "Science [MULT] AND ncbijournals [FILT]"
```

with the query producing:

```plaintext
J. Zhejiang Univ. Sci. 1009-3095|Print 1009-3095|Linking
Science (80- ) 0193-4511|Print 0193-4511|Linking
Science 0036-8075|Print 1095-9203|Electronic ...
```

An alias file can also be read in a separate instruction at the beginning of a pipeline or script:

```plaintext
eproxy -alias q_aliases
```
For maximum flexibility, separate eproxy commands can be piped together to load multiple shortcut files, as long as the shortcut strings are all unique.

**E Search and E Link Options**

E Search can be given a -sort argument to specify the order of results when the records are retrieved:

```
esearch -db pubmed -query "opsin gene conversion" -sort "last author" |
    efetch -format docsum |
    xtract -pattern DocumentSummary -element Id LastAuthor PubDate Title
```

E Link can return links to the citation list using "-name pubmed_pubmed_citedin", but only for publications with full text deposited in PubMed Central (PMC). For example, the query:

```
esearch -db pubmed -query "Beadle GW [AUTH]" |
    elink -related -name pubmed_pubmed_citedin |
    efetch -format docsum |
    xtract -pattern Author -element Name |
    sort-uniq-count-rank |
    head -n 10
```

produces a ranked list of the ten most cited authors:

13    Beadle GW  
8     Ephrussi B  
8     Glass NL   
7     Hawley RS  
7     Mitchell MB 
7     PERKINS DD 
7     Tatum EL   
6     Mitchell HK 
6     YANOFSKY C 
5     Langley CH 

Similarly, "-name pubmed_pubmed.refs" returns an article's reference list, again for publications deposited in PMC.

E Link has several command modes, and these can be specified with the -cmd argument. When not using the default "neighbor_history" command, elink will return an E LinkResult XML object, with the links for each UID presented in separate blocks. For example:

```
esearch -db pubmed -query "Hoffmann PC [AUTH] AND dopamine [MAJR]" |
    elink -related -cmd neighbor |
    xtract -pattern LinkSetDb -element Id
```

will show the original PMID in the first column and related article PMIDs in subsequent columns:

1504781    11794494    3815119    1684029    14614914    12128255    ... 
1684029    3815119    1504781    8097798    17161385    14755628    ... 
2572612    2903614    6152036    2905789    9483560     1352865     ... 
...
When the elink command "prlinks" is used with "ref" mode, it can obtain HTML containing or referencing full text articles directly from the publishers. The UNIX "xargs" command calls elink separately for each identifier:

```
epost -db pubmed -id 22966225,19880848 |
efilter -query "free full text [FILT]" |
efetch -format uid |
xargs -n 1 elink -db pubmed -cmd prlinks -mode ref -http get -id
```

The elink -batch flag will bypass the Entrez history mechanism for large queries.

**Xtract Special Topics**

Self-closing tags of the standard form:

```
<Na-strand/>
```

or alternative form:

```
<Na-strand></Na-strand>
```

have no text content and thus cannot be selected with an -element command. Instead, use -match to detect the desired object and -lbl to print a specified string:

```
-match Na-strand -lbl "Na-strand"
```

If the tag contains an attribute:

```
<Seq-interval_strand>
  <Na-strand value="plus"/>
</Seq-interval_strand>
```

it can be selected by matching on the specified value:

```
-group Seq-interval_strand \
  -block Seq-interval_strand -match Na-strand@value:plus -lbl "+" \ 
  -block Seq-interval_strand -match Na-strand@value:minus -lbl "-"
```

The -pattern, -group, -block, and -subset commands provide a nested hierarchy of loop organizers for exploration of XML objects. Each pattern can contain multiple groups, each group can encompass multiple blocks, and each block can have multiple subsets.

Use of different argument names allows a linear representation of loop nesting, and provides sufficient flexibility to identify and extract arbitrary data from XML records in Entrez.

Sketching in pseudo code can clarify relative nesting levels. The extraction command:

```
xtract -pattern PubmedArticle \
  -block Author -element Initials,LastName \ 
  -block MeshHeading \
  -match QualifierName \
    -element DescriptorName \ 
    -subset QualifierName -element QualifierName
```
could be represented as a computer program in pseudo code by:

```plaintext
for each Pubmed record {
    for each Author {
        print Initials LastName
    }
    for each MeSH term {
        if Subheadings are present {
            print Term Name
            for each Subheading {
                print Subheading Name
            }
        }
    }
}
```

Extra arguments (-division, -branch, -section, and -unit) are held in reserve to provide additional levels of organization, should the need arise in the future for processing complex, deeply-nested XML data. The full set of commands, in order of rank, are:

- pattern
- division
- group
- branch
- block
- section
- subset
- unit

Starting xtract exploration with -block, and expanding with -group and -subset, leaves additional level names that can be used wherever needed without having to redesign the entire command.

Using "xtract -synopsis | sort-uniq-count" instead of -outline produces a table of unique path counts:

```
1    DbBuild
1    DocumentSummary
1    DocumentSummary/ArticleIds
5    DocumentSummary/ArticleIds/ArticleId
5    DocumentSummary/ArticleIds/ArticleId/IdType
5    DocumentSummary/ArticleIds/ArticleId/IdTypeN
5    DocumentSummary/ArticleIds/ArticleId/Value
1    DocumentSummary/Attributes
1    DocumentSummary/Authors
2    DocumentSummary/Authors/Author
2    DocumentSummary/Authors/Author/AuthType
...  
```

**Heterogeneous Data**

XML objects can contain a heterogeneous mix of components. For example:
returns a mixture of book and journal records:

```xml
<PubmedArticleSet>
  <PubmedBookArticle>
    <BookDocument>
      ...
    </PubmedBookData>
  </PubmedBookArticle>
  <PubmedArticle>
    <MedlineCitation>
      ...
    </PubmedData>
  </PubmedArticle>
</PubmedArticleSet>
```

The "Parent/*" construct is used to visit the individual components, even though they may have different names. Piping the XML output to:

```
xtract -pattern "PubmedArticleSet/*" -element ***
```

separately prints the entirety of each XML component:

```
<PubmedBookArticle><BookDocument> ...
</PubmedBookData></PubmedBookArticle>
<PubmedArticle><MedlineCitation> ...
</PubmedData></PubmedArticle>
```

Use of the "Parent/Child" construct can isolate objects of the same name that differ by their location in the XML hierarchy. For example:

```
efetch -db pubmed -id 21433338,17247418 -format xml |
xtract -pattern "PubmedArticleSet/"
  -group "BookDocument/AuthorList" -tab "\\n" -element LastName \
  -group "Book/AuthorList" -tab "\\n" -element LastName \
  -group "Article/AuthorList" -tab "\\n" -element LastName
```

writes separate lines for book/chapter authors, book editors, and article authors:

```
Fauci        Desrosiers
Coffin       Hughes        Varmus
Lederberg    Cavalli       Lederberg
```

Simply exploring with individual arguments:

```
-g group BookDocument -block AuthorList -element LastName
```

would visit the editors (at BookDocument/Book/AuthorList) as well as the authors (at BookDocument/AuthorList), and print names in order of appearance in the XML:

```
Coffin Hughes Varmus Fauci Desrosiers
```
(In this particular example the book author lists could be distinguished by using -match "@Type:authors" or -match "@Type:editors", but exploring by "Parent/Child" is a general position-based approach.)

**Recursive Definitions**

Certain XML objects returned by efetch are recursively defined, including Taxon in TaxaSet (-db taxonomy) and Gene-commentary in Entrezgene_comments (-db gene). Thus, they can have nested objects with the same XML tag.

Retrieving a set of taxonomy records:

```
efetch -db taxonomy -id 9606,7227 -format xml
```

produces XML with nested Taxon objects (marked below with line references) for each rank in the taxonomic lineage:

```
<TaxaSet>
  1  <Taxon>
      <TaxId>9606</TaxId>
      <ScientificName>Homo sapiens</ScientificName>
      ...<LineageEx>
  2  <Taxon>
      <TaxId>131567</TaxId>
      <ScientificName>cellular organisms</ScientificName>
      <Rank>no rank</Rank>
  3  </Taxon>
  4  <Taxon>
      <TaxId>2759</TaxId>
      <ScientificName>Eukaryota</ScientificName>
      <Rank>superkingdom</Rank>
  5  </Taxon>
      ...<LineageEx>
  ...<Taxon>
  6  <Taxon>
  7  <Taxon>
      <TaxId>7227</TaxId>
      <ScientificName>Drosophila melanogaster</ScientificName>
      ...<LineageEx>
  8  </Taxon>
</TaxaSet>
```

Although <Taxon> on line 1 is actually closed by </Taxon> on line 6, use of "-pattern Taxon" to visit data between <Taxon> and </Taxon> pairs will incorrectly match it with the first <Taxon> on line 3.

Xtract circumvents the nesting artifact with an alternative search algorithm that tracks XML object depth, instead of using regular expression pattern matching. This is selected by using capitalized versions of the exploration commands:
Extraction of data with a capitalized -Pattern argument:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml |
xtract -Pattern Taxon \ 
    -first TaxId ScientificName GenbankCommonName Division
```

behaves as desired and returns information for the main entries:

<table>
<thead>
<tr>
<th>TaxId</th>
<th>ScientificName</th>
<th>GenbankCommonName</th>
<th>Division</th>
</tr>
</thead>
<tbody>
<tr>
<td>9606</td>
<td>Homo sapiens</td>
<td>human</td>
<td>Primates</td>
</tr>
<tr>
<td>7227</td>
<td>Drosophila melanogaster</td>
<td>fruit fly</td>
<td>Invertebrates</td>
</tr>
<tr>
<td>10090</td>
<td>Mus musculus</td>
<td>house mouse</td>
<td>Rodents</td>
</tr>
</tbody>
</table>

Use of lower-case "-pattern" erroneously visits the entries for each taxonomic lineage level (skipping the first because of the pattern matching artifact):

<table>
<thead>
<tr>
<th>TaxId</th>
<th>ScientificName</th>
</tr>
</thead>
<tbody>
<tr>
<td>9606</td>
<td>Homo sapiens</td>
</tr>
<tr>
<td></td>
<td>human</td>
</tr>
<tr>
<td></td>
<td>Primates</td>
</tr>
<tr>
<td>2759</td>
<td>Eukaryota</td>
</tr>
<tr>
<td>33154</td>
<td>Opisthokonta</td>
</tr>
<tr>
<td>33208</td>
<td>Metazoa</td>
</tr>
</tbody>
</table>

The "/Child" construct will skip past the first tag (e.g., <Taxon>) so that subsequent explorations are able to visit the internal objects:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml |
xtract -Pattern Taxon -block "/Taxon" \
    -tab "\n" -first TaxId,ScientificName
```

That command returns:

<table>
<thead>
<tr>
<th>TaxId</th>
<th>ScientificName</th>
</tr>
</thead>
<tbody>
<tr>
<td>131567</td>
<td>cellular organisms</td>
</tr>
<tr>
<td>2759</td>
<td>Eukaryota</td>
</tr>
<tr>
<td>33154</td>
<td>Opisthokonta</td>
</tr>
<tr>
<td>33208</td>
<td>Metazoa</td>
</tr>
</tbody>
</table>

Similarly, use of a capitalized -Block argument:

```
efetch -db gene -id 837025,837031 -format xml |
xtract -pattern Entrezgene_comments \ 
    -Block Gene-commentary \ 
        -match "Gene-commentary_heading:Related Sequences" \ 
        -element Gene-commentary_accession
```

"Entrez Direct: E-utils on the UNIX Command Line"
correctly finds all appropriate accessions in Entrezgene records:

DQ446230    ABE65601    CP002684    AEE27818
CP002684    AEE27823    CP002684    AEE27824    CP002684    AEE27825

(The alternative search algorithm in the Perl version of xtract is significantly slower than the regular expression method, so capitalized exploration arguments should only be used when processing recursively-defined objects. The new xtract, written in the Go programming language, has no such performance penalty.)

**Querying External Web Services**

The EDirect inquire function can be used to obtain data from an arbitrary URL. Queries are built up from command-line arguments. For example:

```
  -db pubmed -term insulin
```

reads the URL and then tag/value pairs to generate an E-utilities query:

```
```

Paths can be separated into components, which are combined with slashes, so:

```
```

is converted to:

```
```

Multiple values between tags are combined with commas. Thus:

```
-db nuccore -id U54469 V00328 -rettype fasta
```

is transformed into:

```
db=nuccore&id=U54469,V00328&rettype=fasta
```

A value that starts with a hyphen (or minus sign) can be distinguished from a tag by prefixing it with a backslash, so:

```
nquire -url http://api.geonames.org/countryCode -lat 41.796 -lng "\-87.577"
```

will be sent as:

```
http://api.geonames.org/countryCode?lat=41.796&lng=-87.577
```

and will return "US" for coordinates within Chicago, which has a negative (western hemisphere) longitude value.

The -alias argument can read a file of shortcut keywords and URL aliases. The following aliases are always available:
so the command:

```
nquire -url "(${eutils_url})" esearch.fcgi \  
   -db gds -term "GSE22309 [ACCN] AND gse [ETYP]" -retmax 200
```

will run an ESearch query and return an eSearchResult XML object.

Raw XML with inconsistent line-wrapping and indentation can be reformatted for easier visual inspection of the data structure and content by piping it through:

```
  xtract -format
```

## Automation

### Entrez Direct Commands Within Scripts

Taking an adventurous plunge into the world of programming, a shell script can be written when each output line of one step needs to be processed independently, instead of output being piped in its entirety to the next command. (The simplest shell script is merely a copy of a set of commands that are typed into the terminal for execution.)

In scripts, variables can be set to the results of a command by enclosing the statements in backtick (``'``) characters. The variable name is prefixed by a dollar sign (``$``) to use its value as an argument in another command. Comments start with a pound sign (``#``) and are ignored. Quotation marks within quoted strings are entered by "escaping" with a backslash (``\``). Subroutines can be used to collect common code or simplify the organization of the script.

For example, executing a script file containing:

```
#!/bin/bash -norc

parse_fields() {
  echo "$1" | 
  xtract -pattern Field \  
  -pfx "[" -sfx "]" -element Name \  
  -pfx "" -sfx "" -element FullName Description | 
  sort -t $'	' -k 2,2f | column -s $'	' -t
}

dbs=`einfo -dbs | xtract -pattern DbName -element DbName | sort`

for db in $dbs
do
  eix=`einfo -db $db`
  flds=`parse_fields "$eix"`

  echo "$db"
  echo ""
  echo "$flds"
```

Entrez Programming Utilities Help

Entrez Direct: E-utilities on the UNIX Command Line
will obtain the list of Entrez databases:

assembly
bioproject
biosample
biosystems
...

and then return the abbreviations, names, and descriptions of indexed search fields, for each individual database:

... epigenomics

[ACCN] Accession Accession number of sequence
[ALL] All Fields All terms from all searchable fields
[AUTH] Author Author
[CDAT] Create Date CreateDate
[DOCT] Document Type DocType
[FILT] Filter Limits the records
[KYWD] Keyword Keyword
[ORGN] Organism scientific and common names of organism
[PRID] Project ID ProjectId
[TXID] Taxonomy ID TaxId
[WORD] Text Word Text
[TITL] Title Title
[UID] UID Unique number assigned to publication
...

The shell script command:

```bash
sleep 1
```

adds a one second delay between steps in a loop, and can be used to help prevent overuse of the Entrez servers by advanced scripts.

**Xargs/Sh Loop**

Writing a script to loop through data can sometimes be avoided by creative use of the UNIX `xargs` and `sh` commands. Within the "sh -c" command string, the last name and initials arguments (passed in pairs by "xargs -n 2") are substituted at the "$0" and "$1" variables. All of the commands in the sh string are run separately on each name:

```bash
echo "Garber ED Casadaban MJ Mortimer RK" |
xargs -n 2 sh -c 'esearch -db pubmed -query "$0 $1 [AUTH]" | xtract -pattern ENTREZ_DIRECT -lbl "$1 $0" -element Count'
```

This produces PubMed article counts for each author:
While Loop

A "while" loop can also be used to independently process lines of data. Given a file "organisms.txt" containing genus-species names, the UNIX "cat" command:

```
cat organisms.txt
```

writes the contents of the file:

```
Arabidopsis thaliana
Caenorhabditis elegans
Danio rerio
Drosophila melanogaster
Escherichia coli
Homo sapiens
Mus musculus
Saccharomyces cerevisiae
```

This can be piped to a loop that reads one line at a time:

```bash
while read org
do
  esearch -db taxonomy -query "$org [LNGE] AND family [RANK]" < /dev/null |
  efetch -format docsum |
  xtract -pattern DocumentSummary -lbl "$org" |
  -element ScientificName Division
done
```

looking up the taxonomic family name and BLAST division for each organism:

```
Arabidopsis thaliana        Brassicaceae          eudicots
Caenorhabditis elegans     Rhabditidae           nematodes
Danio rerio                Cyprinidae            bony fishes
Drosophila melanogaster    Drosophilidae         flies
Escherichia coli           Enterobacteriaceae    enterobacteria
Homo sapiens               Hominidae             primates
Mus musculus               Muridae               rodents
Saccharomyces cerevisiae   Saccharomycetaceae    ascomycetes
```

(The "< /dev/null" input redirection construct prevents esearch from "draining" the remaining lines from stdin.)

For Loop

The same results can be obtained with organism names embedded in a "for" loop:

```bash
for org in  
  "Arabidopsis thaliana" 
  "Caenorhabditis elegans" 
```

Entrez Direct: E-utilities on the UNIX Command Line
"Danio rerio" \\n"Drosophila melanogaster" \\n"Escherichia coli" \\n"Homo sapiens" \\n"Mus musculus" \\n"Saccharomyces cerevisiae"

do
  esearch -db taxonomy -query "$org [LNGE] AND family [RANK]" |
  efetch -format docsum |
  xtract -pattern DocumentSummary -lbl "$org" |
  -element ScientificName Division
done

File Exploration

A for loop can also be used to explore the computer's file system:

```
for i in *
  do
    if [ -f "$i" ]
      then
        echo $(basename "$i")
      fi
  done
```

visiting each file within the current directory. The asterisk ("*" character indicates all files, and can be replaced by any pattern (e.g., "*.txt") to limit the file search. The if statement "-f" operator can be changed to "-d" to find directories instead of files, and "-s" selects files with size greater than zero.

Processing in Groups

Because of technical limits in the Entrez link server, it may be necessary to perform an elink operation on a large set of records by using a function that splits unique identifiers or sequence accession numbers into smaller groups:

```
JoinIntoGroupsOf() {
  xargs -n "$@" echo |
  sed 's/ /,;/g'
}
alias join-into-group-of='JoinIntoGroupsOf'
```

The following example will process sequence records in groups of 200 accessions at a time:

```
... efetch -format acc |
join-into-groups-of 200 |
xargs -n 1 sh -c 'epost -db nuccore -format acc -id "$0" |
elink -target pubmed |
efetch -format abstract'
```
Examples

Additional examples of using EDirect to answer impromptu questions are shown in this section.

Author Frequency

Who are the most prolific authors on rattlesnake phospholipase?

```
esearch -db pubmed -query "crotalid venoms [MAJR] AND phospholipase [TIAB]" | efetch -format xml | xtract -pattern PubmedArticle \ -block Author -sep " " -tab "\n" -element LastName,Initials | sort-uniq-count-rank
```

This search produces:

74    Lomonte B
73    Gutiérrez JM
49    Soares AM
48    Marangoni S
43    Giglio JR
39    Bon C
...

Publication Distribution

When were the most papers about Legionnaires disease published?

```
esearch -db pubmed -query "legionnaires disease [TITL]" | efetch -format docsum | xtract -pattern DocumentSummary -element PubDate | cut -c 1-4 | sort-uniq-count-rank
```

reports the number of selected papers per year:

173    1979
102    1980
96     1978
92     1981
66     1983
...

Treatment Locations

What is the geographic distribution of sepsis treatment studies?

```
esearch -db pubmed -query "sepsis/therapy [MESH] AND geographic locations [MESH]" | efetch -format xml | xtract -pattern PubmedArticle \ -block MeshHeading -match "DescriptorName@Type:Geographic"
```
returns the number of articles ranked by country (or region) of study:

<table>
<thead>
<tr>
<th>Country</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>567</td>
</tr>
<tr>
<td>Spain</td>
<td>207</td>
</tr>
<tr>
<td>Great Britain</td>
<td>176</td>
</tr>
<tr>
<td>Germany</td>
<td>156</td>
</tr>
<tr>
<td>India</td>
<td>123</td>
</tr>
<tr>
<td>Europe</td>
<td>118</td>
</tr>
<tr>
<td>France</td>
<td>113</td>
</tr>
<tr>
<td>Taiwan</td>
<td>100</td>
</tr>
<tr>
<td>Japan</td>
<td>89</td>
</tr>
<tr>
<td>Thailand</td>
<td>83</td>
</tr>
<tr>
<td>Italy</td>
<td>75</td>
</tr>
<tr>
<td>England</td>
<td>74</td>
</tr>
</tbody>
</table>

Research History

What is the historic pattern of publication on diphtheria, pertussis, and tetanus?

```
result=""
for disease in diphtheria pertussis tetanus
do
citations=`esearch -db pubmed -query "disease [TITL]"`
current=`for (( yr = 2010; yr >= 1900; yr -= 10 ))
do
  echo "Scitations"
  efilter -mindate "$yr" -maxdate "$(yr+9)" -datetype PDAT |
  xtract -pattern ENTREZ_DIRECT -lbl "$yr:s" -element Count
done`
heading=`echo -e "disease:0:4" | tr [a-z] [A-Z]`
current=`echo -e "Years\tHeading\n-----\t----
$current"`
if [ -n "$result" ]
then
  result=`join -t $'	' <(echo "$result") <(echo "$current")`
else
  result=$current
fi
done
echo "$result"
gives per-decade counts of relevant papers for each disease:

<table>
<thead>
<tr>
<th>Years</th>
<th>DIPH</th>
<th>PERT</th>
<th>TETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010s</td>
<td>408</td>
<td>1173</td>
<td>659</td>
</tr>
<tr>
<td>2000s</td>
<td>890</td>
<td>1967</td>
<td>1341</td>
</tr>
<tr>
<td>1990s</td>
<td>1149</td>
<td>2660</td>
<td>1614</td>
</tr>
<tr>
<td>1980s</td>
<td>780</td>
<td>1746</td>
<td>1485</td>
</tr>
<tr>
<td>1970s</td>
<td>749</td>
<td>698</td>
<td>1523</td>
</tr>
</tbody>
</table>

Entrez Direct: E-utils on the UNIX Command Line
<table>
<thead>
<tr>
<th>Decade</th>
<th>Count</th>
<th>Freq</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960s</td>
<td>1152</td>
<td>635</td>
<td>2086</td>
</tr>
<tr>
<td>1950s</td>
<td>1224</td>
<td>490</td>
<td>1532</td>
</tr>
<tr>
<td>1940s</td>
<td>452</td>
<td>172</td>
<td>237</td>
</tr>
<tr>
<td>1930s</td>
<td>157</td>
<td>26</td>
<td>46</td>
</tr>
<tr>
<td>1920s</td>
<td>128</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>1910s</td>
<td>83</td>
<td>7</td>
<td>41</td>
</tr>
<tr>
<td>1900s</td>
<td>93</td>
<td>3</td>
<td>28</td>
</tr>
</tbody>
</table>

### Protein Homolog

Is there a mammalian equivalent of lycopene cyclase?

```bash
esearch -db protein -query "lycopene beta cyclase [PROT] AND tomato [ORGN]" | elink -related | efetch -format gpc | xtract -pattern INSDSeq -element INSDSeq_division | sort-uniq-count-rank
```

In the resulting list of GenBank division codes:

- 905  BCT
- 856  ENV
- 609  PLN
- 197  CON
- 127  PAT
- 2    SYN

there are no similar sequences (protein neighbors) in the HUM, PRI, ROD, MAM, VRT, or INV divisions, so lycopene cyclase is not present in animals.

### Longest Sequences

What are the longest known insulin precursor molecules?

```bash
esearch -db protein -query "insulin [PROT]" | efetch -format docsum | xtract -pattern DocumentSummary -element Caption Slen Title | grep -v receptor | sort -k 2,2nr | head -n 5 | cut -f 1 | xargs -n 1 sh -c 'efetch -db protein -id "$0" -format gp > "$0".gpf'
```

Post-processing excludes the longer "insulin-like receptor" sequences and saves the GenPept results to individual files named by their sequence accessions:

- EFN61235.gpf
- EFN80340.gpf
- EGW08477.gpf
- EKC18433.gpf
- ELK28555.gpf

using the right angle bracket (">") UNIX output redirection character.
**Archaea Enzyme**

Which archaea bacteria have chloramphenicol acetyltransferase?

```bash
esearch -db protein -query "chloramphenicol acetyltransferase [PROT] AND archaea [ORGN]" |
efetch -format gpc |
xtract -pattern INSDSeq -element INSDSeq_organism INSDSeq_definition |
grep -i chloramphenicol |
cut -f 1 |
sort -f |
uniq
```

produces a list of organism names:

Methanobrevibacter ruminantium
Methanobrevibacter smithii
Methanosarcina acetivorans
...

**Structural Similarity**

What archaea structures are similar to snake venom phospholipase?

```bash
esearch -db structure -query "crotalus [ORGN] AND phospholipase A2" |
elink -related |
efilter -query "archaea [ORGN]" |
efetch -format docsum |
xtract -pattern DocumentSummary |
   -match "PdbClass:Hydrolase" |
   -element PdbDescr |
sort -f |
uniq -i
```

This query uses geometric comparison (structure neighboring) to find proteins that are too divergent to be detected by sequence similarity with a BLAST search:

Crystal Structure Of Autoprocessed Form Of Tk-Subtilisin
Crystal Structure Of Ca2 Site Mutant Of Pro-S324a
Crystal Structure Of Ca3 Site Mutant Of Pro-S324a
...

**Taxonomy Search**

Which organisms contain an annotated RefSeq genome MatK gene?

```bash
esearch -db nuccore -query "MatK [GENE] AND NC_0:NC_999999999 [PACC]" |
efetch -format docsum |
xtract -pattern DocumentSummary -element TaxId |
sort -n |
uniq |
epost -db taxonomy |
efetch -format docsum |
xtract -pattern DocumentSummary -element ScientificName |
sort
```

The first query obtains taxonomy UIDs from nucleotide document summaries and uploads them for separate retrieval from the taxonomy database:
Acidosasa purpurea
Acorus americanus
...
Zingiber spectabile
Zygnema circumcarinatum

Exon Counts

How many exons are in each dystrophin transcript variant?

```bash
esearch -db gene -query "DMD [GENE] AND human [ORGN]" |
efetch -format docsum |
xtract -pattern DocumentSummary -block GenomicInfoType -tab "\n" -element ChrAccVer,ChrStart,ChrStop |
```

This search returns the chromosome accession and the (0-based) gene start and stop positions:

```
NC_000023.11    33339608    31119221
```

The values are adjusted by an awk command:

```bash
awk -F'\t' -v 'OFS="\t"' '{if ($2 < $3) 
    {print $1, $2+1, $3+1, 1} else {print $1, $3+1, $2+1, 2}}' |
```

to produce (1-based) coordinates in numerical order, along with a strand parameter:

```
NC_000023.11    31119222    33339609    2
```

These are then passed as arguments to efetch:

```bash
xargs -n 4 sh -c 'efetch -db nuccore -format gbc -id "$0" -seq_start "$1" -seq_stop "$2" -strand "$3"' |
```

which retrieves an INSDSeq XML subset record for the indicated region. That contains a number of alternatively-spliced dystrophin mRNA and CDS features.

Data extraction computes the number of intervals for each mRNA location (corresponding to individual exons or UTRs), and obtains the transcript sequence accession, transcript length, and product name from qualifiers:

```bash
xtract -insd complete mRNA "#INSDInterval" 
    transcript_id "#transcription" product |
```

Final processing sorts by number of exons:

```bash
grep -i dystrophin |
    sed 's/dystrophin, transcript variant //g' |
sort -k 2,2nr -k 4,4nr |
```

resulting in a table of exon counts and transcript lengths:
### Genome Range

What genes are in a given range on the human Y chromosome?

```bash
esearch -db gene -query "Homo sapiens [ORGN] AND Y [CHR]*" | 
efilter -query "alive [PROP]" | 
efetch -format docsum | 
xtract -pattern DocumentSummary -NAME Name \ 
   -block GenomicInfoType -match "ChrLoc:Y" \ 
   -tab "\n" -element "&NAME",ChrAccVer,ChrStart,ChrStop | 
awk -F \t -v 'OFS=\t' '{if ($3 < $4) \ 
  (print $1, $2, $3+1, $4+1, 1) else (print $1, $2, $4+1, $3+1, 2))' | 
sort -t $'\t' -k 3,3n -k 4,4n | 
awk -F \t -v 'OFS=\t' '/^ASMT\t/{a++}/^IL3RA\t/{a++}a>0{print}a>1{exit}'
```

This query returns a table of gene names and sequence locations, for the human "Y" chromosome, in the region between the ASMT and IL3RA genes:

<table>
<thead>
<tr>
<th>Gene</th>
<th>ChrAccVer</th>
<th>ChrStart</th>
<th>ChrStop</th>
<th>Chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL3RA</td>
<td>NC_000024.10</td>
<td>1336601</td>
<td>1382689</td>
<td>1</td>
</tr>
<tr>
<td>LOC102724575</td>
<td>NC_000024.10</td>
<td>1365567</td>
<td>1368017</td>
<td>1</td>
</tr>
<tr>
<td>LOC101928032</td>
<td>NC_000024.10</td>
<td>1378130</td>
<td>1382392</td>
<td>2</td>
</tr>
<tr>
<td>SLC25A6</td>
<td>NC_000024.10</td>
<td>1386152</td>
<td>1392146</td>
<td>2</td>
</tr>
<tr>
<td>ASMTL-AS1</td>
<td>NC_000024.10</td>
<td>1400531</td>
<td>1415421</td>
<td>1</td>
</tr>
<tr>
<td>ASMTL</td>
<td>NC_000024.10</td>
<td>1403139</td>
<td>1453794</td>
<td>2</td>
</tr>
<tr>
<td>P2RY8</td>
<td>NC_000024.10</td>
<td>1462572</td>
<td>1537144</td>
<td>2</td>
</tr>
<tr>
<td>AKAP17A</td>
<td>NC_000024.10</td>
<td>1591593</td>
<td>1602520</td>
<td>1</td>
</tr>
<tr>
<td>ASMT</td>
<td>NC_000024.10</td>
<td>1595455</td>
<td>1643081</td>
<td>1</td>
</tr>
</tbody>
</table>

(The awk command on the last line can be replaced by:

```bash
awk -F \t -v 'OFS=\t' '{if ( 1336601 <= $3 && $4 <= 1643081 ) print}'
```

to search for genes by numeric range.)

(The "ChrLoc:Y" match is necessary because certain genes (e.g., IL9R) are present in the pseudoautosomal regions common to both X and Y chromosomes:

```
...<GenomicInfo>
  <GenomicInfoType>
    <ChrLoc>Y</ChrLoc>
    <ChrAccVer>NC_000024.10</ChrAccVer>
    <ChrStart>5184100</ChrStart>
    <ChrStop>5197336</ChrStop>
```
with each gene copy annotated in its own GenomicInfoType block.)

**Gene Counts**

How many genes are on each human chromosome?

```bash
for chr in {1..22} X Y MT
do
esearch -db gene -query "Homo sapiens [ORG] AND $chr [CHR]" |
efilter -query "alive [PROP] AND genetype protein coding [PROP]" |
efetch -format docsum |
xtract -pattern DocumentSummary -NAME Name \
-block GenomicInfoType -match "ChrLoc:$chr" \
-tab "\n" -element ChrLoc,"&NAME" |
grep "." | sort | uniq | cut -f 1 |
sort-uniq-count-rank
done
```

returns a count of unique protein-coding genes per chromosome:

<table>
<thead>
<tr>
<th>Chr</th>
<th>Gene Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2062</td>
</tr>
<tr>
<td>2</td>
<td>1268</td>
</tr>
<tr>
<td>3</td>
<td>1082</td>
</tr>
<tr>
<td>4</td>
<td>766</td>
</tr>
<tr>
<td>5</td>
<td>872</td>
</tr>
<tr>
<td>6</td>
<td>1035</td>
</tr>
<tr>
<td>7</td>
<td>931</td>
</tr>
<tr>
<td>8</td>
<td>683</td>
</tr>
<tr>
<td>9</td>
<td>800</td>
</tr>
<tr>
<td>10</td>
<td>750</td>
</tr>
<tr>
<td>11</td>
<td>1291</td>
</tr>
<tr>
<td>12</td>
<td>1034</td>
</tr>
<tr>
<td>13</td>
<td>334</td>
</tr>
<tr>
<td>14</td>
<td>612</td>
</tr>
<tr>
<td>15</td>
<td>609</td>
</tr>
<tr>
<td>16</td>
<td>844</td>
</tr>
<tr>
<td>17</td>
<td>1195</td>
</tr>
<tr>
<td>18</td>
<td>275</td>
</tr>
<tr>
<td>19</td>
<td>1407</td>
</tr>
<tr>
<td>20</td>
<td>550</td>
</tr>
<tr>
<td>21</td>
<td>245</td>
</tr>
</tbody>
</table>
The range construct cannot be used for Roman numerals, so the equivalent query on Saccharomyces cerevisiae would need to explicitly list all chromosomes:

```plaintext
for chr in I II III IV V VI VII VIII IX X XI XII XIII XIV XV XVI MT
```

Plastid genes can be selected with "source plastid [PROP]".

## Complete Genomes

What complete genomes are available for Escherichia coli?

```plaintext
esearch -db assembly -query "Escherichia coli [ORGN] AND representative [PROP]" |
  elink -target nuccore -name nuccore_refseq |
  efetch -format docsum |
  xtract -pattern DocumentSummary -element AccessionVersion Slen Title |
  sed 's/,.*//g' |
  sort -t $'	' -k 2,2nr
```

This search finds genomic assemblies and sorts the results by sequence length, allowing complete genomes to be easily distinguished from smaller plasmids:

<table>
<thead>
<tr>
<th>Accession</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC_002695.1</td>
<td>5498450</td>
<td>Escherichia coli O157:H7 str. Sakai chromosome</td>
</tr>
<tr>
<td>NC_018658.1</td>
<td>5273097</td>
<td>Escherichia coli O104:H4 str. 2011C-3493 ...</td>
</tr>
<tr>
<td>NC_011751.1</td>
<td>5202090</td>
<td>Escherichia coli UMN026 chromosome</td>
</tr>
<tr>
<td>NC_011750.1</td>
<td>5132068</td>
<td>Escherichia coli IAI39 chromosome</td>
</tr>
<tr>
<td>NC_017634.1</td>
<td>4747819</td>
<td>Escherichia coli O83:H1 str. NRG 857C chromosome</td>
</tr>
<tr>
<td>NC_000913.3</td>
<td>4641652</td>
<td>Escherichia coli str. K-12 substr. MG1655</td>
</tr>
<tr>
<td>NC_017659.1</td>
<td>147060</td>
<td>Escherichia coli O83:H1 str. NRG 857C plasmid ...</td>
</tr>
</tbody>
</table>

The sed command removes extraneous text in the title (e.g., complete genome, complete sequence, primary assembly) after a comma.

A similar query for humans, additionally filtering out scaffolds, contigs, and plasmids:

```plaintext
esearch -db assembly -query "Homo sapiens [ORGN] AND representative [PROP]" |
  elink -target nuccore -name nuccore_refseq |
  efetch -format docsum |
  xtract -pattern DocumentSummary -element AccessionVersion Slen Title |
  sed 's/,.*//g' | grep -v scaffold | grep -v contig | grep -v plasmid | sort
```

returns the assembled chromosome and mitochondrial sequence records:

<table>
<thead>
<tr>
<th>Accession</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC_000001.11</td>
<td>248956422</td>
<td>Homo sapiens chromosome 1</td>
</tr>
<tr>
<td>NC_000002.12</td>
<td>242193529</td>
<td>Homo sapiens chromosome 2</td>
</tr>
<tr>
<td>NC_000003.12</td>
<td>198295559</td>
<td>Homo sapiens chromosome 3</td>
</tr>
<tr>
<td>Accession</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>NC_000004.12</td>
<td>190214555</td>
<td>Homo sapiens chromosome 4</td>
</tr>
<tr>
<td>NC_000005.10</td>
<td>181538259</td>
<td>Homo sapiens chromosome 5</td>
</tr>
<tr>
<td>NC_000006.12</td>
<td>170805979</td>
<td>Homo sapiens chromosome 6</td>
</tr>
<tr>
<td>NC_000007.14</td>
<td>159345973</td>
<td>Homo sapiens chromosome 7</td>
</tr>
<tr>
<td>NC_000008.11</td>
<td>145138636</td>
<td>Homo sapiens chromosome 8</td>
</tr>
<tr>
<td>NC_000009.12</td>
<td>138394717</td>
<td>Homo sapiens chromosome 9</td>
</tr>
<tr>
<td>NC_000010.11</td>
<td>133797422</td>
<td>Homo sapiens chromosome 10</td>
</tr>
<tr>
<td>NC_000011.10</td>
<td>135086622</td>
<td>Homo sapiens chromosome 11</td>
</tr>
<tr>
<td>NC_000012.12</td>
<td>133275309</td>
<td>Homo sapiens chromosome 12</td>
</tr>
<tr>
<td>NC_000013.11</td>
<td>114364328</td>
<td>Homo sapiens chromosome 13</td>
</tr>
<tr>
<td>NC_000014.9</td>
<td>107043718</td>
<td>Homo sapiens chromosome 14</td>
</tr>
<tr>
<td>NC_000015.10</td>
<td>101991189</td>
<td>Homo sapiens chromosome 15</td>
</tr>
<tr>
<td>NC_000016.10</td>
<td>90338345</td>
<td>Homo sapiens chromosome 16</td>
</tr>
<tr>
<td>NC_000017.11</td>
<td>83257441</td>
<td>Homo sapiens chromosome 17</td>
</tr>
<tr>
<td>NC_000018.10</td>
<td>80373285</td>
<td>Homo sapiens chromosome 18</td>
</tr>
<tr>
<td>NC_000019.10</td>
<td>58617616</td>
<td>Homo sapiens chromosome 19</td>
</tr>
<tr>
<td>NC_000020.11</td>
<td>64444167</td>
<td>Homo sapiens chromosome 20</td>
</tr>
<tr>
<td>NC_000021.9</td>
<td>46709983</td>
<td>Homo sapiens chromosome 21</td>
</tr>
<tr>
<td>NC_000022.11</td>
<td>50818468</td>
<td>Homo sapiens chromosome 22</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>156040895</td>
<td>Homo sapiens chromosome X</td>
</tr>
<tr>
<td>NC_000024.10</td>
<td>57227415</td>
<td>Homo sapiens chromosome Y</td>
</tr>
<tr>
<td>NC_012920.1</td>
<td>16569</td>
<td>Homo sapiens mitochondrion</td>
</tr>
</tbody>
</table>

This process can be automated to loop through a list of specified organisms:

```bash
for org in ""Agrobacterium tumefaciens" "Bacillus anthracis" "Escherichia coli" "Neisseria gonorrhoeae" "Pseudomonas aeruginosa" "Shigella flexneri" "Streptococcus pneumoniae"
  do
esearch -db assembly -query "$org [ORGN]
  dofilter-query "representative [PROP]
  delink -target nuccore -name assembly_nuccore_refseq
  efetch -format docsum
  xtract -pattern DocumentSummary -element AccessionVersion Slen Title
  sed 's/,.*//' | grep -v -i -e scaffold -e contig -e plasmid -e sequence -e patch
  sort -t $'	' -k 2,2nr
done
```

which generates:

<table>
<thead>
<tr>
<th>Accession</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC_011985.1</td>
<td>4005130</td>
<td>Agrobacterium radiobacter K84 chromosome 1</td>
</tr>
<tr>
<td>NC_011983.1</td>
<td>2650913</td>
<td>Agrobacterium radiobacter K84 chromosome 2</td>
</tr>
<tr>
<td>NC_005945.1</td>
<td>5228663</td>
<td>Bacillus anthracis str. Sterne chromosome</td>
</tr>
<tr>
<td>NC_003997.3</td>
<td>5227293</td>
<td>Bacillus anthracis str. Ames chromosome</td>
</tr>
<tr>
<td>NC_002695.1</td>
<td>5498450</td>
<td>Escherichia coli O157:H7 str. Sakai chromosome</td>
</tr>
<tr>
<td>NC_018658.1</td>
<td>5273097</td>
<td>Escherichia coli O104:H4 str. 2011C-3493 ...</td>
</tr>
</tbody>
</table>
Amino Acid Composition

What is the amino acid composition of human titin?

```
abbrev=( Ala Asx Cys Asp Glu Phe Gly His Ile \ 
        Xle Lys Leu Met Asn Py1 Pro Gln Arg \ 
        Ser Thr Sec Val Trp Xxx Tyr Glx )
```

efetch -db protein -id "Q8WZ42.4" -format gpc |
xtract -pattern INSDSeq -element INSDSeq_sequence |
fold -w 1 |
sort-uniq-count |
while read num lttr
do
    idx=`printf %i "$lttr"` |
    echo -e "${abbrev[\$idx-97]}\t\$num"
done |
sort
```
produces a table of residue counts using the three-letter amino acid abbreviations:

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ala</td>
<td>2084</td>
</tr>
<tr>
<td>Arg</td>
<td>1640</td>
</tr>
<tr>
<td>Asn</td>
<td>1111</td>
</tr>
<tr>
<td>Asp</td>
<td>1720</td>
</tr>
<tr>
<td>Cys</td>
<td>513</td>
</tr>
<tr>
<td>Gln</td>
<td>942</td>
</tr>
<tr>
<td>Glu</td>
<td>3193</td>
</tr>
<tr>
<td>Gly</td>
<td>2066</td>
</tr>
<tr>
<td>His</td>
<td>478</td>
</tr>
<tr>
<td>Ile</td>
<td>2062</td>
</tr>
<tr>
<td>Leu</td>
<td>2117</td>
</tr>
<tr>
<td>Lys</td>
<td>2943</td>
</tr>
<tr>
<td>Met</td>
<td>398</td>
</tr>
<tr>
<td>Phe</td>
<td>908</td>
</tr>
<tr>
<td>Pro</td>
<td>2517</td>
</tr>
<tr>
<td>Ser</td>
<td>2463</td>
</tr>
<tr>
<td>Thr</td>
<td>2546</td>
</tr>
<tr>
<td>Trp</td>
<td>466</td>
</tr>
<tr>
<td>Tyr</td>
<td>999</td>
</tr>
<tr>
<td>Val</td>
<td>3184</td>
</tr>
</tbody>
</table>
## Amino Acid Substitutions

What are the missense products of green-sensitive opsin?

```bash
esearch -db gene -query "CBD [GENE] AND human [ORGN]" |
ealink -target SNP |
efetch -format xml |
xtract -pattern Rs -RSID Rs@rsId \  
  -block FxnSet -match @fxnClass:missense \  
  -sep "." -ACCN "@protAcc,protVer" \  
  -element "&RSID" "&ACCN" @aaPosition \  
  -tab "/n" -element @residue |
sort -t $'	' -k 2,2 -k 3,3n -k 4,4 | uniq |
awk -F '$'	' -v 'OFS='	' '{print $1, $2, $3+1, $4}'
```

The query returns a table of non-synonymous amino acid substitutions derived from single nucleotide polymorphisms:

<table>
<thead>
<tr>
<th>RSID</th>
<th>ACCESSION</th>
<th>Position</th>
<th>Residue</th>
</tr>
</thead>
<tbody>
<tr>
<td>104894915</td>
<td>NP_000504.1</td>
<td>94</td>
<td>K</td>
</tr>
<tr>
<td>200470120</td>
<td>NP_000504.1</td>
<td>111</td>
<td>I</td>
</tr>
<tr>
<td>201569525</td>
<td>NP_000504.1</td>
<td>116</td>
<td>S</td>
</tr>
<tr>
<td>267606927</td>
<td>NP_000504.1</td>
<td>177</td>
<td>R</td>
</tr>
</tbody>
</table>

The results can be piped to a shell script:

```bash
#!/bin/bash -norc

seq=""
last=""

while read rsid accn pos res
do
  if [ "$accn" != "$last" ]
    then
      insd=`efetch -db protein -id "$accn" -format gbc < /dev/null`
      seq=`echo $insd | xtract -pattern INSDSeq -element INSDSeq_sequence`
      last=$accn
    fi
  echo ">rs$rsid [$accn $res@$pos]"
  if [ $pos -gt 1 ]
    then
      echo ${seq:0:$pos-1} | fold -w 50
    fi
  echo $res
  if [ $pos -lt ${#seq} ]
    then
      echo ${seq:$pos} | fold -w 50
    fi
done
```

to produce protein sequences with the individual residue substitutions in upper case:
Upstream Sequences

What sequences are upstream of phenylalanine hydroxylase genes?

```bash
#!/bin/bash -norc
bases=1500
if [ -n "$1" ]
then
    bases=$1
fi

while read id accn start stop
do
    if [[ $start -eq 0 || $stop -eq 0 || $start -eq $stop ]]
    then
        echo "Skipping $id due to ambiguous coordinates"
        continue
    fi
    if [ $start -gt $stop ]
    then
        stop=$(( start + bases ))
        start=$(( start + 1 ))
    fi
    
```

obtains a series of homologous genes, converting the gene coordinates to 1-based positions suitable for retrieving sequence regions:

<table>
<thead>
<tr>
<th>id</th>
<th>accn</th>
<th>start</th>
<th>stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>5053</td>
<td>NC_000012.12</td>
<td>102917603</td>
<td>102838326</td>
</tr>
<tr>
<td>18478</td>
<td>NC_000076.6</td>
<td>87521795</td>
<td>87584137</td>
</tr>
<tr>
<td>38871</td>
<td>NT_037436.4</td>
<td>7760453</td>
<td>7763166</td>
</tr>
<tr>
<td>24616</td>
<td>NC_005106.4</td>
<td>28066639</td>
<td>28129772</td>
</tr>
<tr>
<td>378962</td>
<td>NC_007115.6</td>
<td>17420391</td>
<td>17402704</td>
</tr>
</tbody>
</table>

Given a shell script named "upstream.sh"
strand=2
else
  stop=$(( start - 1 ))
  start=$(( start - bases ))
  strand=1
fi
rslt=`efetch -db nuccore -id $accn -format fasta \
    -seq_start $start -seq_stop $stop -strand $strand < /dev/null`
  echo "#$rslt"
done

the data lines can be piped through:

upstream.sh 500

to extract and print the 500 nucleotides immediately upstream of each gene. (Without the argument it will default to 1500 nucleotides.)

**Author Combinations**

What are the authorship patterns among selected individuals?

The "coauthors.sh" script takes author name arguments to construct a custom data extraction command for analyzing research collaboration patterns:

```bash
#!/bin/bash -norc
if [ "$#" -lt 2 ]
then
  echo "Must supply at least two author names"
  exit 1
fi
query="xtract -pattern PubmedArticle -element MedlineCitation/PMID"

# append a -block statement for each author argument
for auth in "$@"
  do
    query=`echo "$query -block Author -match "LastName:$auth" -sep " " -element LastName,Initials"`
  done
query=`echo "$query | sort -t \"\t\" -k 2f -k 1,1n"`

if [ -t 0 ]
then
  # stand-alone command, print constructed query for later use
  echo "$query"
else
  # dynamically execute query on XML data piped to script
  res=`eval "$query"`
  echo "$res"
fi
```

*Entrez Direct: E-utilities on the UNIX Command Line*
If XML publication data are piped to the script, it will read the data and immediately execute the generated `xtract` query. Otherwise, if called as a stand-alone command, it will print the custom query instructions for later use.

Running the following command:

```
esearch -db pubmed -query "Casadaban MJ [AUTH] OR Berg CM [AUTH]" | 
efetch -format xml | 
./coauthors.sh Casadaban Groisman Berg Garber | 
./extract-fuse.pl pubmed > author_patterns.htm
```

first produces an internal result table of PMIDs grouped by author combination:

```
... 
7635839  Casadaban MJ
9634770  Casadaban MJ
1827084  Casadaban MJ  Groisman EA
2954879  Casadaban MJ  Groisman EA
3020001  Casadaban MJ  Groisman EA
3525518  Casadaban MJ  Groisman EA
3542967  Casadaban MJ  Groisman EA
6324195  Casadaban MJ  Groisman EA
3301525  Casadaban MJ  Groisman EA  Berg CM
```

The sorted lines are then piped to the "extract-fuse.pl" script:

```
#!/usr/bin/perl

my $max = scalar @ARGV;
if ( $max < 1 ) {
  die "Need argument for database\n";
}
my $db = $ARGV[0];

my $thisline = "";
my $laststr = "";
my $str = "";
my $uid = "";
my $uidlist = "";
my $count = 0;

my $pfx = "";
while ($thisline = <STDIN>) {
  $thisline =~ s/\r//;
  $thisline =~ s/\n//;
  if ($thisline =~ /^([\t\t]+)\t(.+)/) {
    $uid = $1;
    $str = $2;
    if ( lc ($str) ne lc ($laststr) and $laststr ne "" ) {
```

Entrez Direct: E-utilities on the UNIX Command Line
which combines them into PubMed query URLs, one for each author pattern:


Those are then wrapped, along with a record count, in the appropriate HTML tags for web display. If the resulting file is opened with a browser, it presents an argument-order-dependent view of author collaboration:

( 55 ) - Berg CM
( 10 ) - Berg CM, Berg DE
( 1 ) - BERG CM, GARBER ED
( 6 ) - Berg DE, Berg CM
( 39 ) - Casadaban MJ
( 6 ) - Casadaban MJ, Groisman EA
( 1 ) - Casadaban MJ, Groisman EA, Berg CM

Clicking on a hyperlinked record count number opens the document summary or individual article page, so the actual publications can be examined.

Indexed Fields

What date fields are indexed for PubMed?

einfo -db pubmed |
xtract -pattern Field |
   -match "IsDate:Y" -and "IsHidden:N" \

Entrez Direct: E-utils on the UNIX Command Line
This produces a list of field abbreviations and names filtered by index type:

<table>
<thead>
<tr>
<th>Field Abbreviation</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDAT</td>
<td>Date - Completion</td>
</tr>
<tr>
<td>CRDT</td>
<td>Date - Create</td>
</tr>
<tr>
<td>EDAT</td>
<td>Date - Entrez</td>
</tr>
<tr>
<td>MHDA</td>
<td>Date - MeSH</td>
</tr>
<tr>
<td>MDAT</td>
<td>Date - Modification</td>
</tr>
<tr>
<td>PDAT</td>
<td>Date - Publication</td>
</tr>
</tbody>
</table>

**Digital Object Identifiers**

How are digital object identifiers obtained from PubMed articles?

```
esearch -db pubmed -query "Rowley JD [AUTH]" | 
efetch -format xml | 
xtract -pattern PubmedArticle -PMID MedlineCitation/PMID | 
   -block ArticleId -match "@IdType:doi" | 
   -tab '\n' -pfx '<a href="http://dx.doi.org/' | 
   -sep '"'>-sfx '</a></p>' -element ArticleId,"&PMID"
```

extracts the DOIs and constructs the appropriate URL references:

```
<p><a href="http://dx.doi.org/10.1038/leu.2013.340">24496283</a></p>
<p><a href="http://dx.doi.org/10.1073/pnas.1310656110">23818607</a></p>
<p><a href="http://dx.doi.org/10.1073/pnas.1310144110">23798388</a></p>
...
```

These intermediate lines are then piped through:

```
xtract -format \
   -xml '<?xml>' -doctype '<!DOCTYPE html>' \
   -pfx '<html><body>' -sfx '</body></html>'
```

to produce a minimal HTML document with clickable links:

```
<?xml
<!DOCTYPE html>
<html>
<body>
<p>
   <a href="http://dx.doi.org/10.1038/leu.2013.340">24496283</a><br>
</p>
<p>
   <a href="http://dx.doi.org/10.1073/pnas.1310656110">23818607</a><br>
</p>
<p>
   <a href="http://dx.doi.org/10.1073/pnas.1310144110">23798388</a><br>
</p>
...
</body>
</html>
```
Phrase Searching

Can phrase searching be simulated in Entrez?

The "entrez-phrase-search" script takes advantage of the fact that some short phrases are indexed in certain Entrez fields. Given an input phrase, the script generates overlapping pairs of adjacent words, separately queries on each pair to determine which are present in the pubmed title or abstract index, and keeps those that appear in at least 10 articles. Independent phrases are separated by a plus ("+") sign.

For example, passing the following arguments to entrez-phrase-search:

```
selective serotonin reuptake inhibitor + monoamine oxidase inhibitor
```

will generate word pairs from each phrase and run a query on each pair. The individual term counts are:

```
10581    selective serotonin
11113    serotonin reuptake
6205    reuptake inhibitor
13161    monoamine oxidase
3515    oxidase inhibitor
```

The combined query will return a search result with 34 articles, and these can then be retrieved by piping to efetch. The script in its current form will not match phrases with plurals (e.g., serotonin reuptake inhibitors) or hyphens (e.g., monoamine-oxidase inhibitor).

```bash
#!/bin/bash -norc
do_one_query() {
  esearch -db pubmed -query "$*" < /dev/null |
  xtract -pattern ENTREZ_DIRECT -element Count
}

word_at_a_time() {
  sed 's/[^+a-zA-Z0-9]/ /g; s/^ *///' |
  tr 'A-Z' 'a-z' |
  fmt -w 1
}

replace_stop_words() {
  while read line
  do
    case "$line" in
      a | about | again | all | almost | also | although | always | \ 
      among | an | and | another | any | are | as | at | be | \ 
      because | been | before | being | between | both | but | by | \ 
      can | could | did | do | does | done | due | during | each | \ 
      either | enough | especially | etc | for | found | from | \ 
      further | had | has | have | having | here | how | however | \ 
      i | if | in | into | is | it | its | itself | just | kg | km | \ 
      made | mainly | make | may | mg | might | ml | mm | most | 
```

Entrez Direct: E-utilities on the UNIX Command Line
group_phrases() {
uniq | paste -sd "," - |
sed -e 's/+//g' -e 's/+/\+/g' -e 's/,/+/g' -e 's/,+/g' -e 's/+/ /g'
}

word_pairs() {
while read first rest
do
if [ -z "$rest" ]
then
  echo "$first"
  continue
fi
prev=$first
for curr in $rest
do
  echo "$prev $curr"
  prev="$curr"
done
done
}

group_phrase_counts() {
  echo "$*" | word_at_a_time | replace_stop_words | group_phrases | fmt -w 1 | tr ', ' ' ' | word_pairs | while read qry
do
  counts=`do_one_query "$qry" [TIAB]`
done
}
Gene-Protein Links

What proteins are produced by a given gene?

Given a query in the gene database, the following commands:

```bash
esearch -db gene -query "beta galactosidase [PFN]" |
elink -target protein -name gene_protein_refseq -cmd neighbor |
xtract -pattern LinkSet -element Id
```

will show the gene ID in the first column and linked RefSeq protein UIDs in subsequent columns.

Piping the results to a Perl script named "gene-protein-links.pl" will read the identifiers and run separate efetch queries on the gene and protein databases:

```perl
#!/usr/bin/perl

while ($line = <STDIN>) {
    chomp ($line);
    @uids = split( /	/, $line);
    $gene = $uids [0];
    $proteins = join (',', @uids [1..$#uids]);
    $symbol = $data = '';

    $cmd = "efetch -format docsum -db gene -id $gene | ";
    $cmd .= "Xtract -pattern DocumentSummary -element Name CommonName";
    open (CMD, "$cmd") |
    while (<CMD>) {
        $symbol .= $_;
    }
}
```
close CMD;

if ($proteins ne "") {
    $cmd = "efetch -format docsum -db protein -id $proteins | ";
    $cmd .= "xtract -pattern DocumentSummary -element Caption Slen Title";
    open (CMD, "$cmd|");
    while (<CMD>) {
        $data .= $_;
    }
    close CMD;
}

print "$symbol$data\n";
}

printing the gene symbol and organism common name, followed by the protein accessions, lengths, and titles:

<table>
<thead>
<tr>
<th>GLB1</th>
<th>human</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP_001129074</td>
<td>546  beta-galactosidase isoform c preproprotein ...</td>
</tr>
<tr>
<td>NP_001073279</td>
<td>647  beta-galactosidase isoform b [Homo sapiens]</td>
</tr>
<tr>
<td>NP_000395</td>
<td>677  beta-galactosidase isoform a preproprotein ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Glb1</th>
<th>house mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP_033882</td>
<td>647  beta-galactosidase preproprotein [Mus musculus]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Glb1</th>
<th>Norway rat</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP_001101662</td>
<td>647  beta-galactosidase precursor [Rattus norvegicus]</td>
</tr>
</tbody>
</table>

**Bulk Downloads**

How can the entire set of GenBank records for mammals be obtained?

```bash
ftp-ls ftp.ncbi.nih.gov genbank |
grep ".seq.gz" |
grep -e gbmam -e gbpri -e gbrod |
xargs -n 1 |
while read file
do
    ftp-cp ftp.ncbi.nih.gov genbank "$file"
gzcat "$file"
rm "$file"
done
```

will use the ftp-ls and ftp-cp scripts (included with the EDirect software) to retrieve and print GenBank flatfiles for human, primate, rodent, and other mammals:

```
GBMAM1.SEQ       Genetic Sequence Data Bank
February 15 2015

NCBI-GenBank Flat File Release 206.0
```
Other Mammalian Sequences (Part 1)

20709 loci, 155323216 bases, from 20709 reported sequences

**LOCUS** AB000170  2732 bp  mRNA  linear  MAM ...
**DEFINITION** Sus scrofa mRNA for endopeptidase 24.16, complete cds.
**ACCESSION** AB000170
**VERSION** AB000170.1  GI:1783121
**KEYWORDS** endopeptidase 24.16 type M3; endopeptidase 24.16 type M1.
**SOURCE** Sus scrofa (pig)
**ORGANISM** Sus scrofa
  Eukaryota; Metazoa; Chordata; Craniata; Vertebrata;
  Euteleostomi; Mammalia; Eutheria; Laurasiatheria;
  Cetartiodactyla; Suina; Suidae; Sus.
**REFERENCE** 1
**AUTHORS** Kato,A., Sugiura,N., Saruta,Y., Hosoiri,T., Yasue,H. and
  Hirose,S.
**TITLE** Targeting of endopeptidase 24.16 to different subcellular
  compartments by alternative promoter usage
**JOURNAL** J. Biol. Chem. 272 (24), 15313-15322 (1997)
**PUBMED** 9182559
**REFERENCE** 2 (bases 1 to 2732)
**AUTHORS** Hirose,S.
**TITLE** Direct Submission
**JOURNAL** Submitted (27-DEC-1996) Shigehisa Hirose, Tokyo Institute of
  Technology, Department of Biological Sciences; 4259
  Nagatsuta-cho, Midori-ku, Yokohama, Kanagawa 226-8501, Japan
**FEATURES**
  Location/Qualifiers
  source  1..2732
    /organism="Sus scrofa"
    /mol_type="mRNA"
    /db_xref="taxon:9823"
    /tissue_type="Liver"
    /dev_stage="Adult"
    /note="porcine"
  mRNA  1..2732
    /note="corresponding to exon1,5-16 of this gene;
    endopeptidase 24.16 type 1"
  CDS  175..2289
    /standard_name="endopeptidase 24.16"
    /note="oligopeptidase M :neurolysin :sBAP(soluble
    angiotensin-binding protein) :MEP(microsomal
    metalloendopeptidase)"
    /codon_start=1
    /product="endopeptidase 24.16 type M1"
    /protein_id="BAA19060.1"
    /db_xref="GI:1783122"
...
Appendices

Setting Contact Address and Script Name

EDirect automatically obtains the user’s e-mail address from the system, to have someone to notify in case a runaway script causes problems with an Entrez server, but if another contact address is desired (e.g., that of a system administrator or software developer) it can be explicitly set at the beginning of a pipeline or script:

```bash
econtact -email author_email_address -tool name_of_script
```

That way the NCBI has information on who to contact if an infinite loop in a script accidentally abuses NCBI resources. (For convenience, the preferred e-mail address and software tool name can also be set in all E-utilities-calling operations.)

Command-Line Arguments

Arguments for the EDirect functions are listed below:

Use esearch to start a new Entrez search on indexed terms:

**Query Specification**

- `-db` Database name
- `-query` Query string

**Document Order**

- `-sort` Result presentation order

**Date Constraint**

- `-days` Number of days in the past
- `-datetype` Date field abbreviation
- `-mindate` Start of date range
- `-maxdate` End of date range

**Miscellaneous Arguments**

- `-label` Alias for query step

The elink function looks up related articles or associated records:

**Destination Database**

- `-related` Neighbors in same database
- `-target` Links in different database
- `-name` Link name (e.g., `pubmed_protein_refseq`)

**Direct Record Selection**

- `-db` Database name
- `-id` Unique identifier(s)
Advanced Control

- cmd Command type (returns eLinkResult XML)
- mode "ref" uses LinkOut provider's web site
- holding Name of LinkOut provider

Batch Processing

- batch Bypass Entrez history mechanism

Miscellaneous Arguments

- label Alias for query step

Use efilter to restrict search or link results by indexed terms:

Query Specification

- query Query string

Date Constraint

- days Number of days in the past
- datatype Date field abbreviation
- mindate Start of date range
- maxdate End of date range

Miscellaneous Arguments

- label Alias for query step

The record retrieval function is efetch:

Format Selection

- format Format of record or report
- mode text, xml, asn.1, json

Direct Record Selection

- db Database name
- id Unique identifier or accession number

Sequence Range

- seq_start First sequence position to retrieve
- seq_stop Last sequence position to retrieve
- strand Strand of DNA to retrieve
- complexity 0 - default, 1 - bioseq, 3 - nuc-prot set

Gene Range
The `xtract` function is used for processing XML data:

**Exploration Argument Hierarchy**

- **pattern** Name of record within set
- **group** Use of different argument
- **block** names allows command-line
- **subset** control of nested looping

**Conditional Execution**

- **position** Must be at given location in list
- **match** Element [@attribute] [:value] required
- **avoid** Skip if element matches
- **and** All tests must pass
- **or** Any passing test suffices

**Format Customization**

- **ret** Override line break between patterns
- **tab** Replace tab character between fields
- **sep** Separator between group members
- **pfx** Prefix to print before group
- **sfx** Suffix to print after group
- **lbl** Insert arbitrary text

**Item Selection**

- **element** Print all items that match tag name
- **first** Only print value of first item
- **last** Only print value of last item
- **NAME** Record value in named variable

**-element Constructs**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Caption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Initials, LastName</td>
</tr>
<tr>
<td>Parent/Child</td>
<td>MedlineCitation/PMID</td>
</tr>
<tr>
<td>Attribute</td>
<td>DescriptorName@MajorTopicYN</td>
</tr>
<tr>
<td>Object Count</td>
<td>&quot;#Author&quot;</td>
</tr>
<tr>
<td>Item Length</td>
<td>&quot;%Title&quot;</td>
</tr>
<tr>
<td>Variable</td>
<td>&quot;&amp;NAME&quot;</td>
</tr>
</tbody>
</table>

**Exploration Constructs**

- **Object** DateCreated
- **Parent/Child** Book/AuthorList
- **Heterogeneous** PubmedArticleSet/`
- **Recursive** */Taxon

*Entrez Direct: E-utilities on the UNIX Command Line*
Command Generator

-**insd** Generate INSDSeq extraction commands

-**insd** Argument Order

  **Descriptors** INSDSeq_sequence INSDSeq_definition INSDSeq_division
  **Flags** complete or partial [optional]
  **Feature(s)** CDS,mRNA
  **Qualifiers** INSDFeature_key "#INSDInterval" gene product

XML Formatting

-**format** Repair XML format and indentation
-**outline** Display outline of XML structure
-**synopsis** Display count of unique XML paths

Documentation

-**examples** Print examples of EDirect and xtract usage

The einfo function returns information on Entrez indexed fields:

Database Selection

-**db** Database name
-**dbs** Get all database names

Several additional functions are provided by EDirect:

 epost

-**db** Database name
-**id** Unique identifier(s) or accession number(s)
-**format** uid or acc
-**label** Alias for query step

eproxy

-**alias** File of aliases
-**pipe** Read aliases from stdin

econtact

-**email** Contact person's address
-**tool** Name of script or program

nquire

-**http** "get" uses HTTP GET instead of POST
-**url** Base URL for external search
In addition, -email and -tool are available in all E-utilities-calling functions to override default values, -http get will force the use of GET instead of POST, -alias will specify a file of shortcut keywords and query strings or URL sections, and -help will print the list of arguments for each function.

For debugging, -silent will suppress link failure retry messages, -verbose will display the <ENTREZ_DIRECT> field values at each step, -debug will print the internal URL query and XML results of each step, and -base will specify a particular server for quality assurance testing.

**EFetch Formats**

EFetch -format and -mode values for each database are shown below:

<table>
<thead>
<tr>
<th>-db</th>
<th>-format</th>
<th>-mode</th>
<th>Report Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>(all)</td>
<td>docsum</td>
<td>DocumentSummarySet XML</td>
<td></td>
</tr>
<tr>
<td></td>
<td>docsum</td>
<td>DocumentSummarySet JSON</td>
<td></td>
</tr>
<tr>
<td></td>
<td>full</td>
<td>Same as native except for mesh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>uid</td>
<td>Unique Identifier List</td>
<td></td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>Entrez URL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xml</td>
<td>Same as -format full -mode xml</td>
<td></td>
</tr>
</tbody>
</table>

**bioproject**

<table>
<thead>
<tr>
<th>native</th>
<th>BioProject Report</th>
</tr>
</thead>
</table>

| native | xml | RecordSet XML |

**biosample**

<table>
<thead>
<tr>
<th>native</th>
<th>BioSample Report</th>
</tr>
</thead>
</table>

| native | xml | BioSampleSet XML |

**biosystems**

| native | xml | Sys-set XML |

**gds**

| native | xml | RecordSet XML |

| summary | xml | Summary |

**gene**

<table>
<thead>
<tr>
<th>gene_table</th>
<th>Gene Table</th>
</tr>
</thead>
</table>

| native | Gene Report |

| native | Entrezgene ASN.1 |

| native | Entrezgene-Set XML |

| tabular | Tabular Report |

**homologene**

<table>
<thead>
<tr>
<th>alignmentscores</th>
<th>Alignment Scores</th>
</tr>
</thead>
</table>

| fasta | FASTA |

| homologene | Homologene Report |

| native | Homologene List |

| native | HG-Entry ASN.1 |

*Entrez Direct: E-utilities on the UNIX Command Line*
<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>xml</td>
<td>Entrez-Homologene-Set XML</td>
</tr>
<tr>
<td>mesh</td>
<td>full</td>
<td>Full Record</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>MeSH Report</td>
</tr>
<tr>
<td></td>
<td>xml</td>
<td>RecordSet XML</td>
</tr>
<tr>
<td>nlmcatalog</td>
<td>native</td>
<td>Full Record</td>
</tr>
<tr>
<td></td>
<td>xml</td>
<td>NLMCatalogRecordSet XML</td>
</tr>
<tr>
<td>pmc</td>
<td>medline</td>
<td>MEDLINE</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>pmc-articleset XML</td>
</tr>
<tr>
<td>pubmed</td>
<td>abstract</td>
<td>Abstract</td>
</tr>
<tr>
<td></td>
<td>medline</td>
<td>MEDLINE</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>Pubmed-entry ASN.1</td>
</tr>
<tr>
<td></td>
<td>xml</td>
<td>PubmedArticleSet XML</td>
</tr>
<tr>
<td></td>
<td>(sequences)</td>
<td></td>
</tr>
<tr>
<td>acc</td>
<td></td>
<td>Accession Number</td>
</tr>
<tr>
<td>est</td>
<td></td>
<td>EST Report</td>
</tr>
<tr>
<td>fasta</td>
<td></td>
<td>FASTA</td>
</tr>
<tr>
<td>fasta</td>
<td>xml</td>
<td>TinySeq XML</td>
</tr>
<tr>
<td>fasta_cds_aa</td>
<td>xml</td>
<td>FASTA of CDS Products</td>
</tr>
<tr>
<td>fasta_cds_na</td>
<td>xml</td>
<td>FASTA of Coding Regions</td>
</tr>
<tr>
<td>ft</td>
<td></td>
<td>Feature Table</td>
</tr>
<tr>
<td>gb</td>
<td></td>
<td>GenBank Flatfile</td>
</tr>
<tr>
<td>gb</td>
<td>xml</td>
<td>GBSset XML</td>
</tr>
<tr>
<td>gbc</td>
<td>xml</td>
<td>INSDset XML</td>
</tr>
<tr>
<td>gbwithparts</td>
<td></td>
<td>GenBank with Contig Sequences</td>
</tr>
<tr>
<td>gene_fasta</td>
<td></td>
<td>FASTA of Gene</td>
</tr>
<tr>
<td>gp</td>
<td></td>
<td>GenPept Flatfile</td>
</tr>
<tr>
<td>gp</td>
<td>xml</td>
<td>GBSset XML</td>
</tr>
<tr>
<td>gpc</td>
<td>xml</td>
<td>INSDset XML</td>
</tr>
<tr>
<td>gss</td>
<td></td>
<td>GSS Report</td>
</tr>
<tr>
<td>ipg</td>
<td></td>
<td>Identical Protein Report</td>
</tr>
<tr>
<td>ipg</td>
<td>xml</td>
<td>IPGReportSet XML</td>
</tr>
<tr>
<td>native</td>
<td>text</td>
<td>Seq-entry ASN.1</td>
</tr>
<tr>
<td>native</td>
<td>xml</td>
<td>Bioseq-set XML</td>
</tr>
<tr>
<td>seqid</td>
<td></td>
<td>Seq-id ASN.1</td>
</tr>
<tr>
<td>snp</td>
<td>chr</td>
<td>Chromosome Report</td>
</tr>
<tr>
<td>docset</td>
<td></td>
<td>Summary</td>
</tr>
<tr>
<td>fasta</td>
<td></td>
<td>FASTA</td>
</tr>
<tr>
<td>flt</td>
<td></td>
<td>Flat File</td>
</tr>
<tr>
<td>native</td>
<td>asn.1</td>
<td>Rs ASN.1</td>
</tr>
<tr>
<td>native</td>
<td>xml</td>
<td>ExchangeSet XML</td>
</tr>
<tr>
<td>rsr</td>
<td></td>
<td>RS Cluster Report</td>
</tr>
</tbody>
</table>

*Entrez Direct: E-utilities on the UNIX Command Line*
ESearch Sort Order

ESearch -sort values for several databases are listed below:

<table>
<thead>
<tr>
<th>-db</th>
<th>-sort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

epigenomics
Date Created
Date Updated
Default Order

gene
Chromosome
Gene Weight
Name
Relevance

geoprofiles
Default Order
Deviation
Mean Value
Outliers
Subgroup Effect

pubmed
First Author
Journal
Last Author
Pub Date
Recently Added
Relevance
Title

(sequences)
Accession
Date Modified
ELink Commands

ELink -cmd options produce results as LinkSet XML:

<table>
<thead>
<tr>
<th>-cmd</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>neighbor</td>
<td>Neighbors or links</td>
</tr>
<tr>
<td>neighbor_score</td>
<td>Neighbors with computed similarity scores</td>
</tr>
<tr>
<td>acheck</td>
<td>All links available</td>
</tr>
<tr>
<td>ncheck</td>
<td>Existence of neighbors</td>
</tr>
<tr>
<td>lcheck</td>
<td>Existence of external links (LinkOuts)</td>
</tr>
<tr>
<td>llinks</td>
<td>Non-library LinkOut providers</td>
</tr>
<tr>
<td>llinkslib</td>
<td>All LinkOut providers</td>
</tr>
<tr>
<td>prlinks</td>
<td>Primary LinkOut provider, or URL for single UID with -mode ref</td>
</tr>
</tbody>
</table>

EInfo Data

EInfo field data contains status flags for several term list index properties:

```xml
<Field>
  <Name>ALL</Name>
  <FullName>All Fields</FullName>
  <Description>All terms from all searchable fields</Description>
  <TermCount>138982028</TermCount>
  <IsDate>N</IsDate>
  <IsNumerical>N</IsNumerical>
  <SingleToken>N</SingleToken>
  <Hierarchy>N</Hierarchy>
  <IsHidden>N</IsHidden>
  <IsTruncatable>Y</IsTruncatable>
</Field>
```
UNIX Utilities

Several useful classes of UNIX text processing filters, with selected arguments, are presented below:

Process by contents:

**sort**  Sorts lines of text
- `f`  Ignore case
- `n`  Numeric comparison
- `r`  Reverse result order
- `k`  Field key (start, stop or first)
- `u`  Unique lines with identical keys

**uniq**  Removes repeated lines
- `c`  Count occurrences
- `i`  Ignore case
- `f`  Ignore first n fields
- `s`  Ignore first n characters
- `d`  Only output repeated lines
- `u`  Only output non-repeated lines

**grep**  Matches patterns using regular expressions
- `i`  Ignore case
- `v`  Invert search
- `w`  Search expression as a word
- `x`  Search expression as whole line
- `e`  Specify individual pattern
- `c`  Only count number of matches
- `n`  Print line numbers

Regular expressions:

Characters

.  Any single character (except newline)
\w  Alphabetic [A-Za-z], numeric [0-9], or underscore (_)
\s  Whitespace (space or tab)
\ Escapes special characters
[] Matches any enclosed characters

Positions

^ Beginning of line
$ End of line
\b Word boundary

Repeat Matches

? 0 or 1
* 0 or more
+ 1 or more
{n} Exactly n

Modify contents:

sed Replaces text strings
   -e Specify individual expression
tr Translates characters
   -d Delete character
rev Reverses characters on line

Format contents:

column Aligns columns by content width
   -s Specify field separator
   -t Create table
expand Aligns columns to specified positions
   -t Tab positions
fold Wraps lines at a specific width
   -w Line width

Filter by position:

cut Removes parts of lines
   -c Characters to keep
   -f Fields to keep
   -d Specify field separator
   -s Suppress lines with no delimiters
head Prints first lines
-n Number of lines
tail Prints last lines
-n Number of lines

Miscellaneous:

wc Counts words, lines, or characters
-c Characters
-l Lines
-w Words

xargs Constructs arguments
-n Number of words per batch

Directory conventions and arguments for file navigation commands are shown below:

cd Changes directory

/ Root
~ Home
. Current
.. Parent
- Previous

ls Lists file names
-l One entry per line
-a Show files beginning with dot (.)
-l List in long format
-R Recursively explore subdirectories
-S Sort files by size

pwd Prints working directory path

Additional documentation with detailed explanations and examples can be obtained by typing "man" followed by a command name.

Terminal Keyboard Shortcuts

Control and escape sequences can be used within a terminal session to navigate through the command history and to move the cursor for editing the command currently being entered:

Command history:

Ctrl-n Next command
Ctrl-p Previous command
Move cursor forward:

Ctrl-e    To end of line
Ctrl-f    By one character
Esc-f     By one argument

Move cursor backward:

Ctrl-a    To beginning of line
Ctrl-b    By one character
Esc-b     By one argument

Delete:

Del       Previous character
Ctrl-d    Next character
Ctrl-k    To end of line
Ctrl-u    Entire line
Ctrl-w    Previous word
Esc-Del   Previous argument
Esc-d     Next argument

Autocomplete:

Tab       Completes directory or file names

Program control:

Ctrl-c    Quit running program
^x^y      Run last command replacing x with y

(Note that Control sequences are typed by holding down Control, hitting the other key, and releasing Control, while Escape sequences are typed by hitting Escape and then hitting the other key.)

Release Notes

EDirect Version 3.90: March 21, 2016
• Code optimizations for increased Xtract speed.

• Xtract can distribute its work among available processor cores for additional speed.

EDirect Version 3.70: February 8, 2016
• Xtract performance improvements.

EDirect Version 3.60: January 11, 2016
• The setup.sh configuration script now downloads a precompiled Xtract executable for selected platforms.
EDirect Version 3.50: December 27, 2015
• Xtract reports error for element:value construct outside of -match or -avoid arguments.

EDirect Version 3.40: December 20, 2015
• Xtract -insd supports extraction from multiple features (e.g., CDS,mRNA).

EDirect Version 3.30: December 3, 2015
• Efetch -format docsum can accept a single sequence accession number in the -id argument.

EDirect Version 3.20: November 30, 2015
• Xtract supports -match conditional execution on values recorded in variables.

EDirect Version 3.10: November 18, 2015
• Efetch adds -chr_start and -chr_stop arguments to specify sequence range from 0-based coordinates in gene docsum GenomicInfoType object.

EDirect Version 3.00: October 30, 2015
• Xtract rewritten in the Go programming language for speed. The setup.sh configuration script installs an older Perl version (2.99) if a local Go compiler is unavailable.
• Efetch -format docsum only decodes HTML entity numbers in select situations.

EDirect Version 2.90: October 15, 2015
• Xtract warns on use of deprecated arguments -present, -absent, and -trim, in preparation for release of much faster version.

EDirect Version 2.80: September 9, 2015
• Xtract uses the "*/Child" construct for nested exploration into recursive structures, replacing the -trim argument.

EDirect Version 2.70: July 14, 2015
• Added entrez-phrase-search script to query on adjacent word pairs indexed in specific fields.

EDirect Version 2.60: June 23, 2015
• Xtract -match and -avoid support "Parent/Child" construct for BLAST XML.

EDirect Version 2.50: April 9, 2015
• Xtract capitalized -Pattern handles recursively-defined top-level objects.

EDirect Version 2.40: March 25, 2015
• EDirect programs use the http_proxy environment variable to work behind firewalls.

EDirect Version 2.30: March 11, 2015
• Cleaned up logic in setup.sh configuration script.
• EPost -format acc works properly on protein accessions.
EDirect Version 2.20: March 4, 2015

- Xtract -match and -avoid recognize "@attribute" without element or value.

EDirect Version 2.10: February 3, 2015

- Added ftp-ls and ftp-cp scripts for convenient access to the NCBI anonymous ftp server.

EDirect Version 2.00: August 28, 2014

- Introduced copy-and-paste installation commands with setup.sh configuration script.

EDirect Version 1.90: August 8, 2014

- Xtract -format combines multiple XML results into a single valid object.
- Improved suppression of 0-count failure messages with -silent flag in scripts.

EDirect Version 1.80: July 15, 2014

- EPost -format acc accepts accessions in an -id argument on the command line.

EDirect Version 1.70: April 23, 2014

- EFetch -format docsum decodes HTML entity numbers embedded in the text.

EDirect Version 1.60: April 3, 2014

- Minor enhancements to xtract -insd.

EDirect Version 1.50: March 29, 2014

- ESearch -sort specifies the order of results when records are retrieved.
- Xtract exploration arguments (e.g., -block) now work on self-closing tags with data in attributes.

EDirect Version 1.40: March 17, 2014

- Xtract -format repairs XML line-wrapping and indentation.
- Implemented -help flag to display the list of command-line arguments for each function.

EDirect Version 1.30: March 3, 2014

- Xtract -insd partial logic was corrected to examine both 5' and 3' partial flags, and the location indicator recognizes "+" or "complete" and "+" or "partial".

EDirect Version 1.20: February 26, 2014

- Xtract -insd detects if it is part of an EDirect sequence record query, and dynamically executes the extraction request for specific qualifier values. When run in isolation it generates extraction instructions that can be incorporated (with modifications, if necessary) into other queries.

EDirect Version 1.10: February 10, 2014

- ESummary was replaced by "efetch -format docsum" to provide a single command for all document retrieval. The esummary command will continue to work for those who prefer it, and to avoid breaking existing scripts.
• Xtract processes each -pattern object immediately upon receipt, eliminating the need for using xargs and sh to split document retrieval into smaller units.

**EDirect Version 1.00: February 6, 2014**

• Initial public release.

**For More Information**

**Announcement Mailing List**

NCBI posts general announcements regarding the E-utilities to the [utilities-announce announcement mailing list](mailto:utilities-announce@lists.soe.ucsc.edu). This mailing list is an announcement list only; individual subscribers may **not** send mail to the list. Also, the list of subscribers is private and is not shared or used in any other way except for providing announcements to list members. The list receives about one posting per month. Please subscribe at the above link.

**Getting Help**

Please refer to the PubMed and Entrez help documents for more information about search queries, database indexing, field limitations and database content.

Suggestions, comments, and questions specifically relating to the EUtility programs may be sent to **eutilities@ncbi.nlm.nih.gov**.