Entrez Direct: E-utilities on the UNIX Command Line

Jonathan Kans, PhD


Getting Started

Introduction

Entrez Direct (EDirect) provides access to the NCBI’s suite 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 includes 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 \n    'set ftp = new Net::FTP("ftp.ncbi.nlm.nih.gov", Passive => 1);
       ftp->login; ftp->binary;
       ftp->get("/entrez/entrezdirect/edirect.tar.gz");'
  gunzip -c edirect.tar.gz | tar xf -
  rm edirect.tar.gz
  export PATH=$PATH:$HOME/edirect
```

1 NCBI; Email: kans@ncbi.nlm.nih.gov.

Corresponding author.

NLM Citation: Kans J. Entrez Direct: E-utilities on the UNIX Command Line. 2013 Apr 23 [Updated 2017 Aug 10]. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-.
This downloads several scripts into an "edirect" folder in the user's home directory. The setup.sh script then downloads any missing Perl modules, and may print an additional command for updating the PATH environment variable in the user's configuration file. Copy that command, if present, and paste it into the terminal window to complete the installation process. The editing instructions will look something like:

```bash
echo "export PATH=\$PATH:\$HOME/edirect" >> $HOME/.bash_profile
```

The configuration file can instead be modified manually using a text editor, if desired.

**Entrez Direct Functions**

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:

```bash
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 allows 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 links to all protein sequences published in the neighbor articles:

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

The vertical bar pipe symbol also allows the query to continue on the next line.

Retrieving PubMed Reports

Piping PubMed query results to efetch and specifying the "abstract" format:

```
esearch -db pubmed -query "lycopene cyclase" | 
efetch -format abstract
```

returns a set of reports that can be read by a person:

```
... 10. PLoS One. 2013;8(3):e58144. doi: 10.1371/journal.pone.0058144. Epub ...Levels of lycopene β-cyclase 1 modulate carotenoid gene expression and accumulation in Daucus carota. Moreno JC(l), 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:
```
... PMID- 23555569 OWN - NLM
Levels of lycopene beta-cyclase 1 modulate carotenoid gene expression and accumulation in Daucus carota.

Retrieving Sequence Reports

Nucleotide and protein records can be downloaded in FASTA format:

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

which consists of a definition line followed by the sequence:

```plain
>gi|735882|gb|AAA81880.1| lycopene cyclase [Arabidopsis thaliana]
MDTLKTPNKLDFIFQFHFERLCSNPPYPSRVLGVKRAIKIVSSVVSAGAALDLVPETKLEMDF
ELPLYDTKSQVVDLAVGGGPAVLAVQVVSEAGLSIDSPSKLIWPNNYGVWDEFEAMLDDLCLDT
TTWSGAVVYVDEGVKDLRSPKGVRBNKVLKSKMLQKCITNGVFKHQSKVTNVHEEANSTVCSDGKV
QASVVLATGFSRLCQVDKVNPQVYQAGYIAEVDHFPFDVMVEDWRDKHLDSYELKERSKIP
TFAMYMPFSSNRFLEELSTVARPLREMDIKERMARLKHGLINVKRIEEDRCVIFPGPLPVLPQVRV
VGIGTGMVFSTGMYVARTLAAAPIVANAIYRLGSPOSSLRSLGQSLSAEVWRLWPIERRRRQREFFC
FGMDILKLDLDDATRFFADDFDLQPYYHGFTPSSRLPFFLPGPLSSLFNASNTSRLEIIMTKGTVPLA
KMINNLFQDRD
```

Additional FASTA-format variants are fasta_cds_na, fasta_cds_aa, and gene_fasta.

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

```plain
LOCUS AAA81880 501 aa linear PLN
DEFINITION lycopene cyclase [Arabidopsis thaliana].
ACCESSION AAA81880
VERSION AAA81880.1 GI:735882
DBSOURCE locus ATHLYC 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 mdtlktpnkl dffipqfhg ferlcsnppvy psvrlgyvkc raiivsvsv sgasaaldlv
61 petkkenldf elplydtsks qvvdlaivgg gpaglavaqq vseaglsvecs idspklwp
121 nnyyvwddef eamddldcltd ttvsvavyyv degvkkkdrp pygrvnrkklk skmlqkct
181 ngvkfhqskv tnv hsv he ans tvvcsdgvlq qasv vld atag fsrclvqydk pyngvqyqav
241 giiaedgdpf fdvdkgvmd wrd khd syp elkernskip tflyampfss nrifleetsl
301 varpglirmed i germaarlk hlginvkrie edercvipmg gpplvlpqrv vgiggtagmv
361 hpstgygmv tlaaapivan aivrylgspqs nsllrgdqls ae vwr dlwpi errrqr effcc
421 fgmdillklld ldatrrffnda ffdlqhywh gflsrlfpel llvflgsafl shasntsrle
481 mtkgvtvpla kmnlnvqdr d
//
...

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
- [ALL]    All Fields
- [AUTH]   Author
- [FAUT]   Author - First
- [LAUT]   Author - Last
- [PDAT]   Date - Publication
- [FILT]   Filter
- [JOUR]   Journal
- [LANG]   Language
- [MAJR]   MeSH Major Topic
- [SUBH]   MeSH Subheading
- [MESH]   MeSH Terms
- [PTYP]   Publication Type
- [WORD]   Text Word
- [TITL]   Title
- [TIAB]   Title/Abstract
- [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]
- pharmacokinetics [MESH]
- chemically induced [SUBH]
- all child [FILT]
- english [FILT]
- freetext [FILT]
- has abstract [FILT]
- historical article [FILT]
- randomized controlled trial [FILT]
- clinical trial, phase ii [PTYP]
- review [PTYP]

Sequence databases are indexed with a different set of search fields, including:

- [ACCN]    Accession
- [ALL]    All Fields
- [AUTH]   Author
- [GPRJ]   BioProject
- [ECNO]   EC/RN Number
- [FKEY]   Feature key
- [FILT]   Filter
- [GENE]   Gene Name
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]
mammalia [ORGN:noexp]
cds [FKEY]
lacz [GENE]
beta galactosidase [PROT]
protein snp [FILT]
reviewed [FILT]
biomol genomic [PROP]
dbxref flybase [PROP]
gdbiv phg [PROP]
phylogenetic study [PROP]
sequence from mitochondrion [PROP]
src cultivar [PROP]
srccdb refseq validated [PROP]
150:200 [SLEN]
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:

```xml
<ENTREZ_DIRECT>
<Db>structure</Db>
<WebEnv>NCID_1_545606712_172.16.22.25_5555_1348089299_358182861</WebEnv>
<QueryKey>7</QueryKey>
<Count>39</Count>
<Step>5</Step>
</ENTREZ_DIRECT>
```

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
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:

```bash
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:

```
15299880
9235985
9141131
8421693
...
```

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:
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:

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

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":

```
-xtract -pattern DocumentSummary
-xtract -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 and value (-if and -unless) or item location (-position):

```
-xtract -if Source -equals "J Bacteriol"
-xtract -position first
```

These commands are issued immediately after an exploration argument.

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

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

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":

```bash
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:

```xml
<DocumentSummarySet status="OK">
  <DbBuild>Build150407-2207m.3</DbBuild>
  <DocumentSummary>
    <Id>19650888</Id>
    <PubDate>2009 Aug 3</PubDate>
    <EPubDate>2009 Aug 3</EPubDate>
    <Source>BMC Microbiol</Source>
    <Authors>
      <Author>
        <Name>Cano V</Name>
        <AuthType>Author</AuthType>
        <ClusterID></ClusterID>
      </Author>
      <Author>
        <Name>Moranta D</Name>
        <AuthType>Author</AuthType>
        <ClusterID></ClusterID>
      </Author>
      ...
    </Authors>
  </DocumentSummary>
</DocumentSummarySet>
```

Piping the document summary output to:

```bash
xtract -outline
```

will give an indented overview of the XML structure hierarchy:

```
DbBuild
DocumentSummary
  Id
  PubDate
  EPubDate
  Source
  Authors
    Author
      Name
      AuthType
      ClusterID
    Author
      Name
      ...
```
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:

```bash
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
```

returns the PubMed identifier (PMID), first author name, and article title:

```
19650888    Cano V        Klebsiella pneumoniae triggers a cytotoxic ...
19262028    Suto J        Metabolic consequence of congenital asplenia ...
19248821    Fukumoto N    Hypoalgesic behaviors of P/Q-type voltage- ...
18822497    Trishin AV    [Protective activity of secreted proteins of ...  
18582214    Singh A       Generation and characterization of monoclonal ...
...
```

Using xtract "-synopsis" instead of -outline show the full path to each element. Piping those results to "sort-uniq-count" (see below) produces a table of unique path counts.

### Processing Results with UNIX Utilities

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

```bash
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 ...
...
```

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:

```bash
WordAtATime() {
    sed 's/[^a-zA-Z0-9]/ /g; s/^ *//' |
tr 'A-Z' 'a-z' |
fmt -w 1
```
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>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>of</td>
<td>296</td>
</tr>
<tr>
<td>the</td>
<td>175</td>
</tr>
<tr>
<td>transposition</td>
<td>114</td>
</tr>
<tr>
<td>and</td>
<td>102</td>
</tr>
<tr>
<td>in</td>
<td>94</td>
</tr>
<tr>
<td>mu</td>
<td>93</td>
</tr>
<tr>
<td>a</td>
<td>83</td>
</tr>
<tr>
<td>dna</td>
<td>61</td>
</tr>
<tr>
<td>tn3</td>
<td>61</td>
</tr>
<tr>
<td>transposon</td>
<td>55</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>GeneID</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 fields are preceded by the -pfx value and followed by the -sfx value, both of which are initially empty.

Missing data values can be marked by the -def argument. For example:

```
esearch -db pubmed -query "deuteranopia" | efetch -format xml | xtract -pattern PubmedArticle -def "-" \ -first MedlineCitation/PMID Author/LastName Keyword```

inserts a dash in a column where the specified element is missing.

**Pubmed Article XML Records**

The PubmedArticle object (for -db pubmed) has a more detailed structure than the DocumentSummary:

```
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
```
Using this information to craft a new xtract statement:

```bash
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>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23393278</td>
<td>Sabbah       Troje        Gray        Hawryshyn</td>
</tr>
<tr>
<td>20884587</td>
<td>Jordan       Deeb         Bosten      Mollon</td>
</tr>
<tr>
<td>18230593</td>
<td>Koshitaka    Kinoshita    Vorobyev    Arikawa</td>
</tr>
<tr>
<td>17685813</td>
<td>Wachtler     Doi          Lee         Sejnowski</td>
</tr>
<tr>
<td>16086150</td>
<td>Goldsmith    Butler</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:

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

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

```xml
...<AuthorList>
 <Author>
   <LastName>Inamdar</LastName>
   <ForeName>Arati A</ForeName>
   <Initials>AA</Initials>
 </Author>
...</AuthorList>
```
Without being given any guidance about context, an -element statement with "Initials" and "LastName" arguments:

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

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

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

Inserting a -block statement redirects data exploration to present each author one at a time. Subsequent -element statements only see the current author’s values:

```bash
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:

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

results in more desirable formatting of author names:

```
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:

```
efetch -db pubmed -id 6092233,4640931,4296474 -format xml |
xtract -pattern PubmedArticle -element MedlineCitation/PMID  
  -block AuthorList -sep "/" -element LastName "#Author"  
  -block PubDate -sep " " -element Year,Month MedlineDate  
  -block DateCreated -sep "-" -element Year,Month,Day  |
sort -t $'\t' -k 3,3n -k 2,2f
```

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 author count:

```
4296474    Friedmann                        1    1968 Apr        1968-06-05
4640931    Tager/Steiner                    2    1972 Dec        1973-02-15
6092233    Calderon/Contopoulou/Mortimer    3    1984 Jul-Aug    1984-12-13
```

(Note that the PubDate object can exist either in a structured form:

```
<PubDate>
  <Year>1968</Year>
  <Month>Apr</Month>
  <Day>25</Day>
</PubDate>
```

(with the Day field frequently absent), or in a string form:

```
<PubDate>
  <MedlineDate>1984 Jul-Aug</MedlineDate>
</PubDate>
```

but would not contain a mixture of both types, so the directive:

```
-element Year,Month MedlineDate
```

will only contribute a single column to the output.)

Nested Exploration of Subsets Within XML Sets

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

```
...  
  <MeshHeading>
    <DescriptorName>RNA, Messenger</ DescriptorName>
    <QualifierName>genetics</QualifierName>
  </MeshHeading>
```

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
```

**Selection of Attributes**

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

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

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

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

The major topic value is placed before each MeSH term or subheading:

```
6162838
N|Base Sequence
```
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
```

**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 -PMID 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
```

**Variable Initialization**

Variables can be initialized with a literal value in parentheses:

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

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:
All variables are reset when the next record is processed.

Conditional Processing

Xtract provides -if and -unless arguments that filter by element name or name plus data value. For example:

```
esearch -db pubmed -query "Cozzarelli NR [AUTH]" |
efetch -format xml |
xtract -pattern PubmedArticle -if "#Author" -eq 3 |
   -block Author -if LastName -is-not Cozzarelli |
   -sep ", " -tab "\n" -element LastName,Initials |
sort | uniq
```

will select papers with exactly 3 authors and print the coauthor names:

```
Ackerman, RS
Adams, DE
Alexandrov, AI
Arimondo, PB
Bauer, WR
...
```

Multiple conditions are specified with -and and -or commands:

```
-if @score -equals 1 -or @score -starts-with 0.9
```

The -else command can supply alternative -element or -lbl instructions to be run if the condition is not satisfied:

```
-if MapLocation -element MapLocation -else -lbl "\-"
```

Parallel -if and -unless statements can be used to provide a more complex response to alternative conditions that includes nested exploration.

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>
      <INSDReference>
        ...
      </INSDReference>
    </INSDSeq_references>
  </INSDSeq>
</INSDSet>
```

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

```
FEATURES
source
  1..17
  /organism="Conus imperialis"
  /db_xref="taxon:35631"
  /country="Philippines"
Protein
  <1..17
  /product="alpha-conotoxin ImI precursor"
mat_peptide
  5..16
  /product="alpha-conotoxin ImI"
  /note="the C-terminal glycine of the precursor is post translationally removed"
```
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 -if and -equals 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:

```
xtract -pattern INSDSeq -ACCN INSDSeq_accession-version \
   -group INSDFeature -if INSDFeature_key -equals mat_peptide \
   -unless INSDFeature_partial5 -or INSDFeature_partial3 \
   -clr -pfx "\n" -element ";ACCN"
```

Each qualifier then generates custom extraction code that is appended to the growing query. For example:

```
-block INSNDQualifier \
   -if INSNDQualifier_name -equals peptide \
   -element INSNDQualifier_value
```

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

```
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:

```
AGO59814.1    32    del13b conotoxin       DCPTSCPPTCANGWECKGYPVRQHCSGCNH
AAO33169.1    16    alpha-conotoxin GIC    GCCSHPACAGNNQHIC
ADB65788.1    20    conotoxin Cal 16       LEMQGCVNANAKFCGECGR
AAN78128.1    12    alpha-conotoxin ImI    GCCSDPRAWRC
AALP3167.1     31    BeTX toxin           CRAEGTYCENDSQCCLNECCWGGCGHPCRHP
ADB65789.1    20    conotoxin Cal 16       LEMQGCVNANAKFCGEGR
AAN78279.1    21    conotoxin Vx-II        WIDPSHYCCGGGCTDDCVNC
ABW16858.1    15    marmophin             DWEYHAHPKPNWF
...
```

Piping the results to a series of UNIX commands:

```
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 \"\t\" -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>Description</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAN78128.1</td>
<td>alpha-conotoxin ImI</td>
<td>GCCSDPRCAWRC</td>
</tr>
<tr>
<td>AAN78127.1</td>
<td>alpha-conotoxin ImII</td>
<td>ACCSDRRCRWRC</td>
</tr>
<tr>
<td>ADB43130.1</td>
<td>conotoxin Cal 1a</td>
<td>KCCKRHGCHEPCGRK</td>
</tr>
<tr>
<td>ADB43131.1</td>
<td>conotoxin Cal 1b</td>
<td>LCCKRHGCHEPCGRK</td>
</tr>
<tr>
<td>AAO33169.1</td>
<td>alpha-conotoxin GIC</td>
<td>GCCSHPACAGNNQHIC</td>
</tr>
<tr>
<td>ADB43128.1</td>
<td>conotoxin Cal 5.1</td>
<td>DPAPCCQHPITECCRR</td>
</tr>
<tr>
<td>AAD31913.1</td>
<td>alpha A conotoxin Tx2</td>
<td>PECCSHPACNVHDPEICR</td>
</tr>
<tr>
<td>ADB43129.1</td>
<td>conotoxin Cal 5.2</td>
<td>MIQRSQCACV KKNCVHVG</td>
</tr>
<tr>
<td>ADD97803.1</td>
<td>conotoxin Cal 1.2</td>
<td>AGCCPTIMKTCRATNRCR</td>
</tr>
<tr>
<td>ADB65789.1</td>
<td>conotoxin Cal 16</td>
<td>LEMQGCVCNANA KFCGCGR</td>
</tr>
<tr>
<td>AAD31912.1</td>
<td>alpha A conotoxin Tx1</td>
<td>PECCSDPRCNSSHPFELCGGR</td>
</tr>
<tr>
<td>AAN78279.1</td>
<td>conotoxin Vx-II</td>
<td>WIDPSHICCCGGCGTDCVNC</td>
</tr>
<tr>
<td>ADB43125.1</td>
<td>conotoxin Cal 14.2</td>
<td>GCPADCPNTDSNKCSPGFPG</td>
</tr>
<tr>
<td>ADD97802.1</td>
<td>conotoxin Cal 6.4</td>
<td>GCWLCUPAMRCRSGVCHDCPR</td>
</tr>
<tr>
<td>CAH64846.1</td>
<td>four-loop conotoxin</td>
<td>CRPSGPGCGVTSICCGCGRS VGKCT</td>
</tr>
<tr>
<td>AAD31915.1</td>
<td>O-superfamily conotoxin TxO2</td>
<td>CYDGSITSCNTGNQCCSGWCIFVCL</td>
</tr>
<tr>
<td>AAD31916.1</td>
<td>O-superfamily conotoxin TxO3</td>
<td>CYDGGSITSCSGICCCS WSIFVCF</td>
</tr>
<tr>
<td>AAD31920.1</td>
<td>omega conotoxin SVIA mutant 1</td>
<td>CRPSGPGCGVTSICCGCRCGKCT</td>
</tr>
<tr>
<td>AAD31921.1</td>
<td>omega conotoxin SVIA mutant 2</td>
<td>CRPSGPGCGVTSICCGCRCGKCT</td>
</tr>
<tr>
<td>ABE27010.1</td>
<td>conotoxin fe14.1</td>
<td>SPGSTICTMRC RTGN KKFPCNCR</td>
</tr>
<tr>
<td>ABE27011.1</td>
<td>conotoxin fe14.2</td>
<td>SGSTVCKMRCLRYGHLYFSCGCR</td>
</tr>
<tr>
<td>ABE27007.1</td>
<td>conotoxin p114.1</td>
<td>GPGSAICMNRCLGGHYMPFNCN</td>
</tr>
<tr>
<td>ABE27008.1</td>
<td>conotoxin p114.2</td>
<td>GPGSAICMNRCLGGHYMPFCHCR</td>
</tr>
<tr>
<td>ABE27009.1</td>
<td>conotoxin p114.3</td>
<td>GPGSAICMNRCLGGHYMPFCNCD</td>
</tr>
</tbody>
</table>

For records where a particular qualifier is missing:

```bash
esearch -db protein -query "RAG1 [GENE] AND Mus musculus [ORGN]" | efetch -format gpc | xtract -insd source organism strain | sort -t $'\t' -u -k 2,3
```

a dash is inserted as a placeholder:
Sequence Coordinates

Gene Positions

An understanding of sequence coordinate conventions is necessary in order to use gene positions to retrieve the corresponding chromosome subregion with efetch or with the UCSC browser.

Sequence records displayed in GenBank or GenPept formats use a "one-based" coordinate system, with sequence position numbers starting at "1":

```
1    catgccattc    gttgagttgg    aaacaaactt    gccggctagc    cgcatacccg    cggggtggga
61   gaacccggctg    tgtgcggcca    cagccaccat    cctggacaaa    cccggaagcg    tgagtgaggg
121  tcggcgagaa    cttgtgggct    agggtcggac    ctcccaatga    cccgttccca    tccccagggga
181  ccccactccc    ctggtacacct    ctgaccttcg    gtgtcctatc    cttcccttct    agatcccttc
...
```

Under this convention, positions refer to the sequence letters themselves:

```
C    A    T    G    C    C    A    T    T    C
1    2    3    4    5    6    7    8    9   10
```

and the position of the last base or residue is equal to the length of the sequence. The ATG initiation codon above is at positions 2 through 4, inclusive.

For computer programs, however, using "zero-based" coordinates can simplify the arithmetic used for calculations on sequence positions. The ATG codon in the 0-based representation is at positions 1 through 3. (The UCSC browser uses a hybrid, half-open representation, where the start position is 0-based and the stop position is 1-based.)

Software at NCBI will typically convert positions to 0-based coordinates upon input, perform whatever calculations are desired, and then convert the results to a 1-based representation for display. These transformations are done by simply subtracting 1 from the 1-based value or adding 1 to the 0-based value.

Coordinate Conversions

Retrieving the docsum for a particular gene:

```
esearch -db gene -query "BRCA2 [GENE] AND human [ORGN]" |
efetch -format docsum
```

returns the chromosomal position of that gene in 0-based coordinates:

```
...
<GenomicInfoType>
  <ChrLoc>13</ChrLoc>
  <ChrAccVer>NC_000013.11</ChrAccVer>
  <ChrStart>32315479</ChrStart>
  <ChrStop>32399671</ChrStop>
  <ExonCount>27</ExonCount>
```
Piping the document summary to an xtract command:

```
xtract -pattern GenomicInfoType -element ChrAccVer ChrStart ChrStop
```

obtains the accession and 0-based coordinate values:

```
NC_000013.11    32315479    32399671
```

EFetch has `-seq_start` and `-seq_stop` arguments to retrieve a gene segment, but these expect the sequence subrange to be in 1-based coordinates.

To address this problem, two additional efetch arguments, `-chr_start` and `-chr_stop`, allow direct use of the 0-based coordinates:

```
efetch -db nucore -format gb -id NC_000013.11 \
  -chr_start 32315479 -chr_stop 32399671
```

and eliminate the need for writing a UNIX shell command to increment the two values.

Xtract has numeric extraction commands to assist with coordinate conversion. Selecting fields with an `-inc` argument:

```
xtract -pattern GenomicInfoType -element ChrAccVer -inc ChrStart ChrStop
```

obtains the accession and 0-based coordinates, then increments the positions to produce 1-based values:

```
NC_000013.11    32315480    32399672
```

EDirect knows the policies for sequence positions in all relevant Entrez databases (e.g., gene, snp, dbvar), and provides additional shortcuts for converting these to other conventions. For example:

```
xtract -pattern GenomicInfoType -element ChrAccVer -1-based ChrStart ChrStop
```

understands that gene ChrStart and ChrStop fields are 0-based, sees that the desired output is 1-based, and translates the command to convert coordinates using the `-inc` argument. Similarly:

```
-xtract -pattern GenomicInfoType -element ChrAccVer -ucsc-based ChrStart ChrStop
```

leaves the 0-based start value unchanged but increments the original stop value to produce the half-open form that can be passed to the UCSC browser:

```
NC_000013.11    32315479    32399672
```

**Complex Objects**

**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>...
    </BookDocument>
  </PubmedBookArticle>
  <PubmedArticle>
    <MedlineCitation>...
    </MedlineCitation>
  </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:

```
    -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 -if "@Type" -equals authors or -if "@Type" -equals 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 (-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>
  <Taxon>
    <TaxId>9606</TaxId>
    <ScientificName>Homo sapiens</ScientificName>
    ...
    <LineageEx>
      <Taxon>
        <TaxId>131567</TaxId>
        <ScientificName>cellular organisms</ScientificName>
        <Rank>no rank</Rank>
      </Taxon>
      <Taxon>
        <TaxId>2759</TaxId>
        <ScientificName>Eukaryota</ScientificName>
        <Rank>superkingdom</Rank>
      </Taxon>
      ...
    </LineageEx>
    ...
  </Taxon>
  <Taxon>
    <TaxId>7227</TaxId>
    <ScientificName>Drosophila melanogaster</ScientificName>
    ...
  </Taxon>
</TaxaSet>
```

Xtract tracks XML object nesting to determine that the <Taxon> start tag on line 1 is actually closed by the </Taxon> stop tag on line 6, and not by the first </Taxon> encountered on line 3.

When a recursive object is given to an exploration command, selection of data using the -element command:

```
efetch -db taxonomy -id 9606,7227,10090 -format xml |
xtract -pattern Taxon -element TaxId ScientificName GenbankCommonName Division
```
does not examine fields in the internal objects, and returns information only for the main entries:

<table>
<thead>
<tr>
<th>TaxId</th>
<th>ScientificName</th>
<th>CommonName</th>
<th>Phylum</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>

The "*/Child" construct will skip past the outer start tag:

```plaintext
efetch -db taxonomy -id 9606,7227,10090 -format xml |
xtract -pattern Taxon -block "*/Taxon" \
   -tab "\n" -element TaxId,ScientificName
```

to visit the next level of nested objects individually:

<table>
<thead>
<tr>
<th>TaxId</th>
<th>Group</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>
</tbody>
</table>

Recursive objects can be fully explored with a double-star-slash prefix:

```plaintext
esearch -db gene -query "DMD [GENE] AND human [ORGN]" |
  efetch -format gene |
  xtract -pattern Entrezgene -block "**/Gene-commentary" \
     -tab "\n" -element Gene-commentary_type@value,Gene-commentary_accession
```

which visits every child object regardless of nesting depth:

<table>
<thead>
<tr>
<th>Object</th>
<th>Accession</th>
</tr>
</thead>
<tbody>
<tr>
<td>genomic</td>
<td>NC_000023</td>
</tr>
<tr>
<td>mRNA</td>
<td>XM_006724469</td>
</tr>
<tr>
<td>peptide</td>
<td>XP_006724532</td>
</tr>
<tr>
<td>mRNA</td>
<td>XM_011545467</td>
</tr>
<tr>
<td>peptide</td>
<td>XP_011543769</td>
</tr>
</tbody>
</table>

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

```bash
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.

### Additional EDirect Options

ESearch and EFilter can be given a `-sort` argument to specify the order of results when the records are retrieved:

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

ELink 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:

```bash
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:

1. Beadle GW
2. Ephrussi B
3. Glass NL
4. Hawley RS
5. Mitchell MB
6. PERKINS DD
7. Tatum EL
8. Mitchell HK
9. YANOFSKY C
10. Langley CH
Similarly, "-name pubmed_pubmed.refs" returns an article's reference list, again for publications deposited in PMC.

ELink 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 ELinkResult 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  11754494  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 "freetext [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. 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 \n-block Seq-interval_strand -if Na-strand@value -equals plus -lbl "+" \n-block Seq-interval_strand -if Na-strand@value -equals 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 \ 
      -if QualifierName \ 
         -element DescriptorName \ 
            -subset QualifierName -element QualifierName
```

could be represented as a computer program in pseudo code by:

```
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.
Querying External Web Services

The EDirect nquire 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:

```
https://eutils. ... .gov/entrez/eutils/esearch.fcgi?db=pubmed&term=insulin
```

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:
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:

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

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

dbs=`einfo -dbs | sort`

for db in $dbs 
do 
eix=`einfo -db $db`
  flds=`parse_fields "$eix"`
  echo "$db"
  echo ""
  echo "$flds"
  echo ""
  sleep 1 
done
```

will obtain the list of Entrez databases:

```
annotinfo
assembly
bioproject
...
and then return the abbreviations, names, and descriptions of indexed search fields, for each individual database:

... mesh

[ALL] All Fields All terms from all searchable fields
[FILT] Filter Limits the records
[MESH] MeSH Terms MeSH Terms
[MHUI] MeSH Unique ID NLM MeSH Browser Unique ID
[MULT] Multi Multi
[PREV] Previous Indexing Previous Indexing
[TYPE] Record Type Record type
[REG] Registry Number Registry Number
[NEXT] Scope Note Scope Note
[ALSO] See Also See Also
[SUSB] Substance Note Substance Name
[WORD] Text Word Free text
[TN] Tree Number Tree Number
[UID] UID Unique number assigned to publication
...

The shell script command:

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:

```
echo "Ed 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:

```
ED Garber       35
MJ Casadaban    46
RK Mortimer     85
```

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:

```
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:

```
for org in 
  "Arabidopsis thaliana" 
  "Caenorhabditis elegans" 
  "Danio rerio" 
  "Drosophila melanogaster" 
  "Escherichia coli" 
  "Homo sapiens" 
  "Mus musculus" 
  "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:

```bash
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:

```bash
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:

```bash
... 
  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 ad hoc questions are shown in this section.

**Author Frequency**

Who are the most prolific authors on rattlesnake phospholipase?

```bash
esearch -db pubmed -query ""crotalid venoms [MAJR] AND phospholipase [TIAB]" | 
  efetch -format xml | 
  xtract -pattern PubmedArticle \
```
Publication Distribution

When were the most papers about Legionnaires disease published?

```bash
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?

```bash
esearch -db pubmed -query \
  "sepsis/therapy [MESH] AND geographic locations [MESH]" |
efetch -format xml |
xtract -pattern PubmedArticle \
  -block MeshHeading -if DescriptorName@Type -equals Geographic \
  -tab "\n  -element DescriptorName |
sort-uniq-count-rank
```

returns the number of articles ranked by country (or region) of study:

- 567 United States
- 207 Spain
- 176 Great Britain
- 156 Germany
- 123 India
- 118 Europe
- 113 France
- 100 Taiwan

...
89  Japan
83  Thailand
75  Italy
74  England
...

Research History

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

```bash
#!/bin/bash
result=""
for disease in diphtheria pertussis tetanus
do
current="for (( yr = 2010; yr >= 1900; yr -= 10 ))
do
  esearch -db pubmed -query "$disease [TITL] AND $yr:($(yr+9)) [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----\n$current"
if [ -n "$result" ]
  then
    result='join -t $'\t' <(echo "$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>577</td>
<td>1708</td>
<td>914</td>
</tr>
<tr>
<td>2000s</td>
<td>892</td>
<td>1966</td>
<td>1344</td>
</tr>
<tr>
<td>1990s</td>
<td>1150</td>
<td>2661</td>
<td>1615</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>1524</td>
</tr>
<tr>
<td>1960s</td>
<td>1152</td>
<td>635</td>
<td>2086</td>
</tr>
<tr>
<td>1950s</td>
<td>1226</td>
<td>491</td>
<td>1540</td>
</tr>
<tr>
<td>1940s</td>
<td>452</td>
<td>173</td>
<td>239</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]"
```
In the resulting list of GenBank division codes:

<table>
<thead>
<tr>
<th>Division Code</th>
<th>Division</th>
</tr>
</thead>
<tbody>
<tr>
<td>905</td>
<td>BCT</td>
</tr>
<tr>
<td>856</td>
<td>ENV</td>
</tr>
<tr>
<td>609</td>
<td>PLN</td>
</tr>
<tr>
<td>197</td>
<td>CON</td>
</tr>
<tr>
<td>127</td>
<td>PAT</td>
</tr>
<tr>
<td>2</td>
<td>SYN</td>
</tr>
</tbody>
</table>

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?

```
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 archaebacteria have chloramphenicol acetyltransferase?

```
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 \ 
  -if PdbClass -equals 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

**Chromosome Locations**

Where are mammalian calmodulin genes located?

```bash
esearch -db gene -query "calmodulin [PFN] AND mammalia [ORGN]" |
efetch -format docsum |
xtract -pattern DocumentSummary -MAP "(-)" -MAP MapLocation |
  -element Id Name "&MAP" ScientificName
```

...
The MAP variable is initialized with a literal dash to prevent missing data from shifting columns in the table:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>801</td>
<td>CALM1</td>
<td>14q32.11</td>
<td>Homo sapiens</td>
</tr>
<tr>
<td>808</td>
<td>CALM3</td>
<td>19q13.32</td>
<td>Homo sapiens</td>
</tr>
<tr>
<td>805</td>
<td>CALM2</td>
<td>2p21</td>
<td>Homo sapiens</td>
</tr>
<tr>
<td>24242</td>
<td>Calm1</td>
<td>6q32</td>
<td>Rattus norvegicus</td>
</tr>
<tr>
<td>12313</td>
<td>Calm1</td>
<td>12 E</td>
<td>Mus musculus</td>
</tr>
<tr>
<td>326597</td>
<td>CALM</td>
<td>-</td>
<td>Bos taurus</td>
</tr>
<tr>
<td>50663</td>
<td>Calm2</td>
<td>6q12</td>
<td>Rattus norvegicus</td>
</tr>
<tr>
<td>24244</td>
<td>Calm3</td>
<td>1q21</td>
<td>Rattus norvegicus</td>
</tr>
<tr>
<td>12315</td>
<td>Calm3</td>
<td>7 9.15 cM</td>
<td>Mus musculus</td>
</tr>
<tr>
<td>12314</td>
<td>Calm2</td>
<td>17 E4</td>
<td>Mus musculus</td>
</tr>
<tr>
<td>617095</td>
<td>CALM1</td>
<td>-</td>
<td>Bos taurus</td>
</tr>
<tr>
<td>396838</td>
<td>CALM3</td>
<td>6</td>
<td>Sus scrofa</td>
</tr>
</tbody>
</table>

The -else command can also be used to insert placeholders for missing data:

```bash
esearch -db gene -query "calmodulin [PFN] AND mammalia [ORGN]" | efetch -format docsum | xtract -pattern DocumentSummary \  -if MapLocation -element Id Name MapLocation ScientificName \  -else -element Id Name -lbl "-" -element ScientificName
```

The -def command can achieve the same result for missing elements:

```bash
esearch -db gene -query "calmodulin [PFN] AND mammalia [ORGN]" | efetch -format docsum | xtract -pattern DocumentSummary \  -def "-" -element Id Name MapLocation ScientificName
```

**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
```

These are then passed to efetch in (0-based) -chr_start and -chr_stop arguments:

```bash
xargs -n 3 sh -c 'efetch -db nuccore -format gbc \  -id "$0" -chr_start "$1" -chr_stop "$2"'
```

which converts them to (1-based) -seq_start and -seq_stop arguments and 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:

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

Final processing sorts by number of exons:

```
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:

<table>
<thead>
<tr>
<th>Genome Location</th>
<th>Exons</th>
<th>Transcript ID</th>
<th>Transcript Length</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC_000023.11</td>
<td>79</td>
<td>NM_004010.3</td>
<td>14083</td>
<td>Dp427p2</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>79</td>
<td>NM_000109.3</td>
<td>14069</td>
<td>Dp427c</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>79</td>
<td>NM_004009.3</td>
<td>14000</td>
<td>Dp427p1</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>79</td>
<td>NM_004006.2</td>
<td>13993</td>
<td>Dp427m</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>78</td>
<td>XM_006724468.1</td>
<td>13920</td>
<td>X1</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>78</td>
<td>XM_006724469.1</td>
<td>13802</td>
<td>X2</td>
</tr>
<tr>
<td>NC_000023.11</td>
<td>77</td>
<td>XM_006724470.1</td>
<td>13881</td>
<td>X3</td>
</tr>
</tbody>
</table>

**Genome Range**

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

```
esearch -db gene -query "Homo sapiens [ORGN] AND Y [CHR]" | 
  efilter -status alive | efetch -format docsum | 
  xtract -pattern DocumentSummary -NAME Name -DESC Description \
    -block GenomicInfoType -if ChrLoc -equals Y \
    -min ChrStart,ChrStop -element "&NAME" "&DESC" | 
  sort -k 1,1n | cut -f 2- | 
  between-two-genes ASMT IL3RA
```

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

<table>
<thead>
<tr>
<th>Gene</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL3RA</td>
<td>interleukin 3 receptor subunit alpha</td>
</tr>
<tr>
<td>LOC101928032</td>
<td>uncharacterized LOC101928032</td>
</tr>
<tr>
<td>LOC101928055</td>
<td>uncharacterized LOC101928055</td>
</tr>
<tr>
<td>SLC25A6</td>
<td>solute carrier family 25 member 6</td>
</tr>
<tr>
<td>LOC105373102</td>
<td>uncharacterized LOC105373102</td>
</tr>
<tr>
<td>LINC00106</td>
<td>long intergenic non-protein coding RNA 106</td>
</tr>
<tr>
<td>ASMTL-AS1</td>
<td>ASMTL antisense RNA 1</td>
</tr>
<tr>
<td>ASMTL</td>
<td>acetylserotonin O-methyltransferase-like</td>
</tr>
<tr>
<td>P2RY8</td>
<td>purinergic receptor P2Y8</td>
</tr>
<tr>
<td>AKAP17A</td>
<td>A-kinase anchoring protein 17A</td>
</tr>
<tr>
<td>ASMT</td>
<td>acetylserotonin O-methyltransferase</td>
</tr>
</tbody>
</table>

(The "-if ChrLoc -equals Y" test is necessary because certain genes (e.g., IL9R) are present in the pseudoautosomal regions common to both X and Y chromosomes:}
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 [ORGN] AND $chr [CHR]"
   | efilter -query "alive [PROP] AND genetype protein coding [PROP]"
   | efetch -format docsum | xtract -pattern DocumentSummary -NAME Name \
   -block GenomicInfoType -if ChrLoc -equals "$chr"
   -tab "$\n" -element ChrLoc,"\NAME" |
   sort | uniq | cut -f 1 | sort-uniq-count-rank
done
```

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

```
2067    1
1268    2
1071    3
755     4
873     5
1034    6
935     7
690     8
801     9
739     10
1288    11
1027    12
335     13
607     14
608     15
```
The range construct cannot be used for Roman numerals, so the equivalent query on Saccharomyces cerevisiae would need to explicitly list all chromosomes:

```python
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?

```bash
esearch -db assembly -query "Escherichia coli [ORGN] AND representative [PROP]" |
elink -target nuccore -name assembly_nuccore_refseq |
efetch -format docsum |
xtract -pattern DocumentSummary -element AccessionVersion Slen Title |
sed 's/,.*//' |
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:

```
NC_002695.1    5498450    Escherichia coli O157:H7 str. Sakai chromosome
NC_018658.1    5273097    Escherichia coli O104:H4 str. 2011C-3493 ...
NC_011751.1    5202090    Escherichia coli UMN026 chromosome
NC_011750.1    5132068    Escherichia coli IAI39 chromosome
NC_017634.1    4747819    Escherichia coli O83:H1 str. NRG 857C chromosome
NC_000913.3    4641652    Escherichia coli str. K-12 substr. MG1655
NC_017659.1    147060     Escherichia coli O83:H1 str. NRG 857C plasmid ...
```

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:

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

returns the assembled chromosome and mitochondrial sequence records:
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]" |
    efilter -query "representative [PROP]" |
    elink -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:

```plaintext
NC_011985.1  4005130  Agrobacterium radiobacter K84 chromosome 1
NC_011983.1  2650913  Agrobacterium radiobacter K84 chromosome 2
NC_005945.1  5228663  Bacillus anthracis str. Sterne chromosome
NC_003997.3  5227415  Bacillus anthracis str. Ames chromosome
```
Amino Acid Composition

What is the amino acid composition of human titin?

```bash
abbrev=( Ala Asx Cys Asp Glu Phe Gly His Ile \ Xle Lys Leu Met Asn Pyl 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 |
tr A-Z a-z |
sed 's/[^a-z]//g' |
fold -w 1 |
sort-uniq-count |
while read num lttr
do
  idx=$(printf %i "$lttr")
  ofs=$((idx-97))
  echo -e "${abbrev[$ofs]}\t$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>
</tbody>
</table>
Tyr 999  
Val 3184

**Amino Acid Substitutions**

What are the missense products of green-sensitive opsin?

```bash
ApplySNPs() {
  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
      pos=$((pos+1))
      pfx=""
      sfx=""
      echo "$accn $res@$pos"
      if [ $pos -gt 1 ]
        then
          pfx=$(echo ${seq:0:$pos-1})
        fi
      if [ $pos -lt ${#seq} ]
        then
          sfx=$(echo ${seq:$pos})
        fi
      echo "$pfx$res$sfx" | fold -w 50
    done
}
esearch -db gene -query "CBD [GENE] AND human [ORGN]" |
elink -target snp |
efetch -format xml |
xtract -pattern Rs -RSID Rs@rsId |
  -block FxnSet -if @fxnClass -equals missense |
  -sep "." -element "&RSID" @protAcc,@protVer @aaPosition |
  -tab "n" -element @residue |
sort -t $'\t' -k 2,2 -k 3,3n -k 4,4 | uniq | ApplySNPs
```

The query returns an intermediate table of non-synonymous amino acid substitutions (with 0-based location coordinates) derived from single nucleotide polymorphisms:

```
1048994915 NP_000504.1 93 K
782122931  NP_000504.1 95 V
781899063  NP_000504.1 97 T
781807082  NP_000504.1 102 A
...
The rows are then processed to produce protein sequences with the individual residue substitutions in upper case:

```
>rs104894915 [NP_000504.1 K@94]
maqqwslqrlaghpqdsyedstqssiftytnsnstrgpfegpyhiapr
wyhltsvwmifvvlsvtgqlmvlaatmkfkkkrhlplnwilvKlavadl
aetviastisvvnqvgygfvlghpmcvlegytvs1cgitglwslaiswe
...
```

### 3'UTR Sequences
What are the 3' UTR sequences for lycopene cyclase?

```bash
ThreePrimeUTRs() {
    xtract -pattern INSDSeq -ACC INSDSeq_accession-version -SEQ INSDSeq_sequence -group INSDFeature -if INSDFeature_key -equals CDS -PRD "(-)" -block INSDQualifier -if INSDQualifier_name -equals product -PRD INSDQualifier_value -block INSDFeature -pfc "\n" -element "&ACC" -rst -last INSDInterval_to -element "&SEQ" &PRD | while read acc pos seq prd do
        if [ $pos -lt ${#seq} ]
            then
                echo -e ">$acc 3'UTR: $((pos+1))..${#seq} $prd"
                echo "$seq:$pos" | fold -w 50
            else
                echo -e ">$acc NO 3'UTR"
            fi
        done
}
esearch -db nuccore -query "5.5.1.19 [ECNO]" |
efilter -molecule mrna -source refseq |
efetch -format gbc | ThreePrimeUTRs
```

prints the sequences immediately following the CDS stop codon:

```
>NM_001328461.1 3'UTR: 1737..1871 lycopene beta cyclase, chloroplastic
gatgaatatagagttactgtgttgtaagctaattcatcatcactgatgcaag
tgcattatcacatttatctctgtgatgattgttccataagattatgagt
tagccattatatataaaaaaaaaaaaa
>NM_001316759.1 3'UTR: 1628..1690 lycopene beta cyclase, chloroplastic
atccgagtaattcgaatctgatlcttccatcttatagcctatatataatac
...
```

### Upstream Sequences
What sequences are upstream of phenylalanine hydroxylase genes?

```bash
esearch -db nuccore -query "U49897 [ACCN]" |
efilter -target gene |
```
elink -target homologene |
elink -target gene |
efetch -format docsum |
xtract -pattern DocumentSummary -if GenomicInfoType -element Id \
-block GenomicInfoType -element ChrAccVer ChrStart ChrStop |
awk -F 't' -v 'OFS=t' '{print $1, $2, $3+1, $4+1}'

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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":

```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=$(( start + 1 ))
        strand=2
    else
        stop=$(( 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
```

done

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:

```
#!/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 -if LastName -equals "$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
```

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

... 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(.+)$/) {
      $uid = $1;
      $str = $2;

      if ( lc ($str) ne lc ($laststr) and $laststr ne "" ) {
         $laststr =~ s/\t/, /g;
         print "<p><a href="$base/$db/$uidlist">";
         print "$count </a>) - $laststr</p>\n";
         $pfx = "";
         $count = 0;
         $uidlist = "";
      }
      $laststr = $str;

      $uidlist .= "$pfx$uid";
      $pfx = ",";
      $count++;
   }
}

53
if ( $laststr ne "" ) {
    $laststr =~ s/\t/, /g;
    print "<p>(<a href="$base/\$db/\$uidlist">";
    print " $count </a>) - $laststr</p>
};

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 \n   -if IsDate -equals Y -and IsHidden -equals N \n      -pfx "[" -sfx "]" -element Name \n      -pfx "," -sfx "," -element FullName |
sort -k 2f | expand

This produces a list of field abbreviations and names filtered by index type:

[CDAT]  Date - Completion
[CRDT]  Date - Create
[EDAT]  Date - Entrez
[MHDA]  Date - MeSH
[MDAT]  Date - Modification
[PDAT]  Date - Publication
Digital Object Identifiers

How are digital object identifiers obtained from PubMed articles?

```bash
esearch -db pubmed -query "Rowley JD [AUTH]" |
efetch -format xml |
xtract -head '<html><body>' -tail '</body></html>' \
-pattern PubmedArticle -PMID MedlineCitation/PMID \
-block ArticleId -if @IdType -equals doi \
-sep '\n' -pfx '<a href="http://dx.doi.org/" \
-sep '\'> -sfx '</a>' -encode ArticleId,"&PMID"
```

extracts the DOIs and constructs the appropriate URL references:

```html
<html><body>
<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>
...
</body>
</html>
```

These intermediate lines are then piped through:

```bash
xtract -format
```

to produce a minimal HTML document with clickable links:

```xml
<?xml version="1.0"?>
<!DOCTYPE html>
<html>
<body>
<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>
...
</body>
</html>
```

Phrase Searching

Can phrase searching be simulated in Entrez?

The "entrez-phrase-search" script included with EDirect 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, running the following command:

```bash
entrez-phrase-search -db pubmed -field WORD \
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:

- 11343 selective serotonin
- 11892 serotonin reuptake
- 6714 reuptake inhibitor
- 21722 monoamine oxidase
- 3680 oxidase inhibitor

The combined query will return a search result with 36 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).

**Gene-Protein Links**

What proteins are produced by a given gene?

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

- `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:

```
#!/usr/bin/perl

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

    $symbol = $data = '';  # symbol and data
    $cmd = "efetch -format docsum -db gene -id $gene | ";
    $cmd .= "xtract -pattern DocumentSummary -element Name CommonName";
    open (CMD, "$cmd|");  # execute the command
    while (<CMD>) {  # read the output
        $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>) {  # read the output
            $symbol .= $_;
        }
        close CMD;
    }
}
```
while (<CMD>) {
    $data .= $_;
}
close CMD;
}

print "$symbol$data
";
}

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Organism</th>
<th>Accession</th>
<th>Length</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLB1</td>
<td>human</td>
<td>NP_001129074</td>
<td>546</td>
<td>beta-galactosidase isoform c preproprotein</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NP_001073279</td>
<td>647</td>
<td>beta-galactosidase isoform b [Homo sapiens]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NP_000395</td>
<td>677</td>
<td>beta-galactosidase isoform a preproprotein</td>
</tr>
<tr>
<td>Glb1</td>
<td>house mouse</td>
<td>NP_033882</td>
<td>647</td>
<td>beta-galactosidase preproprotein [Mus musculus]</td>
</tr>
<tr>
<td>Glb1</td>
<td>Norway rat</td>
<td>NP_001101662</td>
<td>647</td>
<td>beta-galactosidase precursor [Rattus norvegicus]</td>
</tr>
</tbody>
</table>

**Bulk Downloads**

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

```
grep ".seq.gz" |
grep -e gbmam -e gbpri -e gbrod |
while read file
do
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:

```
LOCUS       AB000170                2732 bp    mRNA    linear   MAM ...  
DEFINITION  Sus scrofa mRNA for endopeptidase 24.16, complete cds.  
```

**Entrez Direct: E-utilities on the UNIX Command Line**
For systems with Aspera Connect installed, the asp-ls and asp-cp scripts can be used for faster retrieval:

```
asp-ls genbank | grep ".seq.gz" | grep -e gbma -e gbpr -e gbod | while read file do
  asp-cp genbank "$file"
gzcat "$file"
```
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

**Limit by Field**

- **-field** Query words individually in field
- **-pairs** Query overlapping word pairs

**Spell Check**

- **-spell** Correct misspellings in query

**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

**Document Order**
- `-sort` Result presentation order

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

**Limit by Field**
- `-field` Query words individually in field
- `-pairs` Query overlapping word pairs

**Spell Check**
- `-spell` Correct misspellings in query
Publication Filters

-pub abstract, clinical, english, free, historical, journal, last_week, last_month, last_year, preprint, review, structured

Sequence Filters

-feature gene, mrna, cds, mat_peptide, ...
-location mitochondrion, chloroplast, plasmid, plastid
-molecule genomic, mrna, trna, rrna, ncrna
-organism animals, archaea, bacteria, eukaryotes, fungi, human, insects, mammals, plants, prokaryotes, protists, rodents, viruses
-source genbank, insd, pdb, pir, refseq, swissprot, tpa

Gene Filters

-status alive
-type coding, pseudo

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

Gene Range

-chr_start Sequence range from 0-based coordinates
-chr_stop in gene docsum GenomicInfoType object

Miscellaneous

-complexity 0 = default, 1 = bioseq, 3 = nuc-prot set
-extend Extend sequence retrieval in both directions
-extrafeat Bit flag specifying extra features
The `xtract` function is used for processing XML data:

**Processing Flags**

- `-compress` Compress runs of spaces
- `-mixed` Allow PubMed mixed content
- `-strict` Remove HTML highlight tags
- `-accent` Delete Unicode accents

**Data Source**

- `-input` Read XML from file instead of stdin

**Exploration Argument Hierarchy**

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

**Exploration Constructs**

- **Object** `DateCreated`
- **Parent/Child** `Book/AuthorList`
- **Heterogeneous** "PubmedArticleSet/*"
- **Nested** "*/Taxon"
- **Recursive** "**/Gene-commentary"

**Conditional Execution**

- `-if` Element [@attribute] required
- `-unless` Skip if element matches
- `-and` All tests must pass
- `-or` Any passing test suffices
- `-else` Execute if conditional test failed
- `-position` Must be at [first|last] location in list

**String Constraints**

- `-equals` String must match exactly
- `-contains` Substring must be present
- `-starts-with` Substring must be at beginning
- `-ends-with` Substring must be at end
- `-is-not` String must not match

**Numeric Constraints**

- `-gt` Greater than
- `-ge` Greater than or equal to
- `-lt` Less than
- `-le` Less than or equal to
-eq Equal to
-ne Not equal to

Format Customization
-rem 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
-clr Clear queued tab separator
-rst Preface combines -cm and -pfx
-def Default placeholder for missing fields
-lbl Insert arbitrary text

Element 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
Tag Caption
Group Initials, LastName
Parent/Child MedlineCitation/PMID
Attribute DescriptorName@MajorTopicYN
Recursive "**/Gene-commentary_accession"
Object Count "#Author"
Item Length "%Title"
Element Depth "^PMID"
Variable "&NAME"

Special -element Operations
-Parent Index "+"
-XML Subtree "*"
-Children "$"
-Attributes "@"

Numeric Processing
-num Count
-len Length
-sum Sum
-min Minimum
-max Maximum
-inc Increment
-dec Decrement
-sub Difference
-avg Average
-dev Deviation

String Processing

-encode URL-encode <, >, &", and ' characters
-upper Convert text to upper-case
-lower Convert text to lower-case
-title Capitalize initial letters of words

Phrase Processing

-terms Partition phrase at spaces
-words Split at punctuation marks
-pairs Adjacent informative words
-letters Separate individual letters
-indices Experimental index generation

Sequence Coordinates

-0-based Zero-Based
-1-based One-Based
-ucsc-based Half-Open

Command Generator

-insd Generate INSDSeq extraction commands

-insd Argument Order

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

Miscellaneous

-head Print before everything else
-tail Print after everything else
-hd Print before each record
-tl Print after each record

Reformatting

-format [copy|compact|flush|indent|expand]

Modification

-filter Object
[retain|remove|encode|decode|shrink|expand|accent]
[content|cdata|comment|object|attributes|container]

Validation
-verify        Report XML data integrity problems

Summary
-outline       Display outline of XML structure
-synopsis      Display count of unique XML paths

Documentation
-examples      Examples of EDirect and xtract usage
-extras        Batch and local processing examples

The einfo function returns information on Entrez indexed fields:

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

Data Summaries
-fields         Print field names
-links          Print link names

Several additional functions are provided by EDirect:
epost
-db             Database name
-id             Unique identifier(s) or accession number(s)
-format         uid or acc
-input          Read from file instead of stdin
-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

qquire
-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>json</td>
<td>DocumentSummarySet JSON</td>
</tr>
<tr>
<td></td>
<td>docsum</td>
<td></td>
<td>Same as native except for mesh</td>
</tr>
<tr>
<td></td>
<td>full</td>
<td></td>
<td>Unique Identifier List</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td></td>
<td>Entrez URL</td>
</tr>
<tr>
<td></td>
<td>xml</td>
<td></td>
<td>Same as -format full -mode xml</td>
</tr>
<tr>
<td>bioproject</td>
<td>native</td>
<td></td>
<td>BioProject Report</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>xml</td>
<td>RecordSet XML</td>
</tr>
<tr>
<td>biosample</td>
<td>native</td>
<td></td>
<td>BioSample Report</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>xml</td>
<td>BioSampleSet XML</td>
</tr>
<tr>
<td>biosystems</td>
<td>native</td>
<td></td>
<td>Sys-set XML</td>
</tr>
<tr>
<td>gds</td>
<td>native</td>
<td></td>
<td>RecordSet XML</td>
</tr>
<tr>
<td></td>
<td>summary</td>
<td>xml</td>
<td>Summary</td>
</tr>
<tr>
<td>gene</td>
<td>gene_table</td>
<td></td>
<td>Gene Table</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td></td>
<td>Gene Report</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>asn.1</td>
<td>Entrezgene ASN.1</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>xml</td>
<td>Entrezgene-Set XML</td>
</tr>
<tr>
<td></td>
<td>tabular</td>
<td></td>
<td>Tabular Report</td>
</tr>
<tr>
<td>homologene</td>
<td>alignmentscores</td>
<td></td>
<td>Alignment Scores</td>
</tr>
<tr>
<td></td>
<td>fasta</td>
<td></td>
<td>FASTA</td>
</tr>
<tr>
<td></td>
<td>homologene</td>
<td></td>
<td>Homologene Report</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td></td>
<td>Homologene List</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>asn.1</td>
<td>HG-Entry ASN.1</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>xml</td>
<td>Entrez-Homologene-Set XML</td>
</tr>
<tr>
<td>mesh</td>
<td>full</td>
<td></td>
<td>Full Record</td>
</tr>
<tr>
<td>Library</td>
<td>Access Type</td>
<td>Format</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>native</td>
<td></td>
<td>xml</td>
<td>MeSH Report</td>
</tr>
<tr>
<td>nlmcatalog</td>
<td></td>
<td></td>
<td>Full Record</td>
</tr>
<tr>
<td>pmc</td>
<td>medline</td>
<td></td>
<td>MEDLINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xml</td>
<td>pmc-articleset XML</td>
</tr>
<tr>
<td>pubmed</td>
<td>abstract</td>
<td></td>
<td>Abstract</td>
</tr>
<tr>
<td></td>
<td>medline</td>
<td></td>
<td>MEDLINE</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>asn.1</td>
<td>Pubmed-entry ASN.1</td>
</tr>
<tr>
<td></td>
<td>native</td>
<td>xml</td>
<td>PubmedArticleSet XML</td>
</tr>
</tbody>
</table>

(sequences)

<table>
<thead>
<tr>
<th>Accession Number</th>
<th>EST Report</th>
<th>FASTA</th>
<th>TinySeq XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>fasta</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fasta_cds_aa</td>
<td></td>
<td>FASTA of CDS Products</td>
<td></td>
</tr>
<tr>
<td>fasta_cds_na</td>
<td></td>
<td>FASTA of Coding Regions</td>
<td></td>
</tr>
<tr>
<td>ft</td>
<td></td>
<td>Feature Table</td>
<td></td>
</tr>
<tr>
<td>gb</td>
<td></td>
<td>GenBank Flatfile</td>
<td></td>
</tr>
<tr>
<td>gb</td>
<td></td>
<td>GBSet XML</td>
<td></td>
</tr>
<tr>
<td>gbc</td>
<td></td>
<td>INSDSet XML</td>
<td></td>
</tr>
<tr>
<td>gbwithparts</td>
<td></td>
<td>GenBank with Contig Sequences</td>
<td></td>
</tr>
<tr>
<td>gene fasta</td>
<td></td>
<td>FASTA of Gene</td>
<td></td>
</tr>
<tr>
<td>gp</td>
<td></td>
<td>GenPept Flatfile</td>
<td></td>
</tr>
<tr>
<td>gp</td>
<td></td>
<td>GBSet XML</td>
<td></td>
</tr>
<tr>
<td>gpc</td>
<td></td>
<td>INSDSet XML</td>
<td></td>
</tr>
<tr>
<td>gss</td>
<td></td>
<td>GSS Report</td>
<td></td>
</tr>
<tr>
<td>ipg</td>
<td></td>
<td>Identical Protein Report</td>
<td></td>
</tr>
<tr>
<td>ipg</td>
<td></td>
<td>IPGReportSet XML</td>
<td></td>
</tr>
<tr>
<td>native</td>
<td>text</td>
<td></td>
<td>Seq-id ASN.1</td>
</tr>
<tr>
<td>native</td>
<td>xml</td>
<td></td>
<td>Bioseq-set XML</td>
</tr>
</tbody>
</table>

.snp

<table>
<thead>
<tr>
<th>Chromosome Report</th>
<th>Summary</th>
<th>FASTA</th>
<th>Flat File</th>
<th>Rs ASN.1</th>
<th>ExchangeSet XML</th>
<th>RS Cluster Report</th>
<th>SS Exemplar List</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>docset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fasta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>native</td>
<td>asn.1</td>
<td></td>
<td></td>
<td></td>
<td>ExchangeSet XML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>native</td>
<td>xml</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RS Cluster Report</td>
<td>SS Exemplar List</td>
</tr>
</tbody>
</table>

.sra

<table>
<thead>
<tr>
<th>EXPERIMENT_PACKAGE_SET XML</th>
<th>SraRunInfo XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>xml</td>
</tr>
</tbody>
</table>
structure

<table>
<thead>
<tr>
<th>mmdb</th>
<th>Ncbi-mime-asnl strucseq ASN.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>MMDB Report</td>
</tr>
<tr>
<td>native</td>
<td>xml</td>
</tr>
<tr>
<td>native</td>
<td>RecordSet XML</td>
</tr>
</tbody>
</table>

taxonomy

<table>
<thead>
<tr>
<th>native</th>
<th>Taxonomy List</th>
</tr>
</thead>
<tbody>
<tr>
<td>native</td>
<td>TaxaSet XML</td>
</tr>
</tbody>
</table>

ESearch Sort Order

ESearch -sort values for several databases are listed below:

- **-db**
- **-sort**

- **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
  - Date Released
  - Default Order
  - Organism Name
  - Taxonomy ID

- **snp**
  - Chromosome Base Position
  - Default Order
  - Heterozygosity
  - Organism
SNP_ID
Success Rate

**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:

```
<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>
  <IsRangable>N</IsRangable>
</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
         -b    Ignore leading blanks
         -s    Stable sort
         -t    Specify field separator
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

File Compression:

tar     Archive files
    -c    Create archive
    -f    Name of output file
    -z    Compress archive with gzip

gzip    Compress file
    -k    Keep original file
    -9    Best compression

unzip   Decompress .zip archive
    -p    Pipe to stdout

gzcat   Decompress .gz archive and pipe to stdout

Directory and File Navigation:

cd      Changes directory

/       Root
~       Home
.       Current
..      Parent
-       Previous

ls       Lists file names
    -l    One entry per line
    -a    Show files beginning with dot (.)
    -1    List in long format
    -R    Recursively explore subdirectories
    -S    Sort files by size
    -t    Sort by most recently modified

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 7.10: August 10, 2017

- Xtract -ascii converts non-ASCII Unicode to hexadecimal numeric character references.
- Setup script recognizes Cygwin running under the MinGW emulator.
**EDirect Version 7.00: July 10, 2017**
- Xtract -mixed and -strict handle multiply-escaped HTML tags.
- Efetch removes normal and escaped HTML tags from PubMed fields.
- Esearch -field processes individual query terms using the designated field, also removing stop words.
- Esearch -pairs splits the query phrase into adjacent overlapping word pairs.

**EDirect Version 6.90: July 5, 2017**
- Xtract -mixed replaces -relaxed, and -accent replaces -plain.
- Efetch uses larger chunks for -format uid, url, and acc.
- Esearch -log shows constructed URL and QueryTranslation result.

**EDirect Version 6.80: June 8, 2017**
- Modified download instructions to use edirect.tar.gz archive.
- The ftp-cp script can now read from stdin without the need for xargs.
- Rerunning ftp-cp or asp-cp only attempts to download missing files.

**EDirect Version 6.70: May 8, 2017**
- Added asp-cp script for faster download of NCBI ftp files using Aspera Connect.
- Xtract -strict and -relaxed handle empty HTML tag variants (e.g., <b/> and <sup/>).

**EDirect Version 6.60: April 25, 2017**
- Xtract -strict replaces -degloss to remove HTML <i>, <b>, <u>, <sup> and <sub> tags from XML contents.
- Xtract -relaxed allows HTML tags in XML contents, to support current PubMed ftp release files.
- Xtract -plain removes Unicode accents.
- The setup.sh script prints an error message if it cannot fetch missing Perl modules.

**EDirect Version 6.50: March 6, 2017**
- Xtract -degloss replaces -html to remove HTML <i>, <b>, <u>, <sup> and <sub> tags.

**EDirect Version 6.40: March 1, 2017**
- Epost detects accession.version input for sequence databases and sets -format acc.
- Xtract -html [remove|encode] converts <i> and <b> tags embedded in XML contents.
EDirect Version 6.30: February 13, 2017
- Efetch -format docsum skips GI-less sequences without summaries.
- Xtract local indexing commands moved to -extras documentation.

EDirect Version 6.20: January 30, 2017
- Xtract -limit and -index allow extraction of selected records from XML file.

EDirect Version 6.10: January 19, 2017
- Added run-ncbi-converter script for processing ASN.1 release files.
- Xtract -format flush option added.
- Removed obsolete accession-dot-version conversion code.

EDirect Version 6.00: December 27, 2016
- Efetch -format docsum removes eSummaryResult wrapper.
- Fixed content truncation bug when Xtract encounters very long sequences.

EDirect Version 5.90: December 21, 2016
- Efetch and Elink readied for switch to accession-dot-version sequence identifier.
- Xtract -insd recognizes INSDInterval_iscomp@value and other boolean attributes.
- Xtract adds experimental phrase processing commands for word index preparation.

EDirect Version 5.80: December 12, 2016
- Efilter adds shortcuts for -db gene (e.g., -status alive, -type coding).
- Xtract numeric conditional tests can use an element name for the second argument (e.g., -if ChrStop -lt ChrStart finds minus strand genes).

EDirect Version 5.70: November 30, 2016
- Xtract -format takes an optional [compact|indent|expand] argument. Processing compact XML is about 15% faster than indent form. Using expand places each attribute on a separate line for ease of reading.

EDirect Version 5.60: November 22, 2016
- Fixed bug in -datetype argument for Esearch and Efilter.
- Added optional argument to filter-stop-words script to indicate replacement.

EDirect Version 5.50: November 16, 2016
- Efetch -id allows non-numeric accessions only for sequence databases.
- Xtract element selection no longer considers fields in recursive sub-objects.
• Xtract introduces a double-star "/Object" construct to flatten recursive child objects for linear exploration.
• Xtract conditional tests ignore empty self-closing tags.
• Xtract -else simplifies insertion of a placeholder to indicate missing data.

**EDirect Version 5.40: November 7, 2016**
• Added filter-stop-words and xy-plot scripts.

**EDirect Version 5.30: October 31, 2016**
• Added support for ecitmatch utility.
• Added amino-acid-composition and between-two-genes scripts.
• The sort-uniq-count and sort-uniq-count-rank scripts take an optional argument (e.g., -n for numeric comparisons, -r to reverse order).

**EDirect Version 5.20: October 26, 2016**
• Setup script no longer modifies the user's configuration file to update the PATH variable. Instead, it now prints customized instructions for the user to execute. The user may choose to run these commands, but is free to edit the .bash_profile file manually.
• Xtract deprecates -match and -avoid functions and the Element:Value conditional shortcut.
• Xtract -if and -unless commands use compound statements for conditional execution (e.g., -if Element -equals Value).
• Colon now separates namespace prefix from element name in xtract arguments (e.g., -block jats:abstract). Colon at start of element name matches any namespace prefix.
• Xtract -insd uses a dash as placeholder for missing field. Experimental -insdx command is deprecated.
• Precompiled versions of xtract are now provided for Darwin, Linux, and CYGWIN_NT platforms. The appropriate executable is downloaded by the setup script.

**EDirect Version 5.10: October 13, 2016**
• Xtract adds -0-based, -1-based, and -ucsc numeric extraction/conversion commands for sequence positions from several Entrez databases.

**EDirect Version 5.00: September 26, 2016**
• Efetch -format fasta removes blank lines between records.
• Xtract -insdx uses a dash to indicate a missing field.
• Xtract -insd no longer has blank lines between records.
• Xtract -input allows reading XML data from a file.
EDirect Version 4.90: September 14, 2016
- Epost -input allows reading from an input file instead of using data piped through stdin.
- Efilter now supports the -sort argument.
- Xtract -filter can recover information in XML comments and CDATA blocks.

EDirect Version 4.80: August 9, 2016
- Xtract -insd controlled vocabularies updated.

EDirect Version 4.70: August 4, 2016
- Einfo -db request can also display -fields and -links data summaries.
- Einfo -dbs prints database names instead of eInfoResult XML.

EDirect Version 4.60: July 18, 2016
- Elink -cmd acheck returns information on all available links for a record.
- Efilter -pub structured limits to articles with structured abstracts.

EDirect Version 4.50: July 1, 2016
- Esearch and Efilter detect and report -query phrase quotation errors.
- Efilter -pub shortcut adds last_week, last_month, and last_year choices.
- Efetch sets -strand 2 for minus strand if -seq_start > -seq_stop or if -chr_start > -chr_stop.

- Transitioning to use of https for access to NCBI services.
- Epost -db assembly -format acc uses [ASAC] field instead of [ACCN].

EDirect Version 4.30: June 13, 2016
- Efilter -pub preprint limits results to ahead-of-print articles.
- Xtract -pattern Parent/* construct can now process catenated XML files.

EDirect Version 4.20: May 24, 2016
- Xtract command-line argument parsing improvements.
- Nquire -get supersedes -http get.

- Xtract -format removes multi-line XML comments and CDATA blocks.
EDirect Version 4.00: April 4, 2016
- Esearch adds -spell to correct known misspellings of biological terms in the query string.
- EfILTER adds -spell to correct query misspellings, and -pub, -feature, -location, -molecule, -organism, and -source shortcuts. Run efILTER -help to see the choices available for each argument.

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

- Xract -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. 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.