Entrez Direct: E-utilities on the UNIX Command Line

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

Getting Started

Introduction

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

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

A laboratory research biologist or medical librarian can use EDirect to build a sophisticated Entrez query without having to write a program, and without needing any formal training in bioinformatics.

EDirect consists of a set of scripts that are downloaded onto the user's computer by anonymous FTP from:

ftp://ftp.ncbi.nih.gov/entrez/entrezdirect/

EDirect commands are actually wrapper scripts that send arguments to the "edirect.pl" program. The scripts must all be kept in the same directory. If that directory is not in the user's "PATH", EDirect commands can be run from within that location by prefixing the program name with the "./" UNIX current directory path symbol. Or the PATH environment variable can be temporarily adjusted to add the entrezdirect directory by starting each terminal session with:

```
PATH=$PATH:$HOME/edirect
```

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

Several examples of ad hoc queries are presented just before the appendices at the end of this document.

Entrez Direct Functions

EDirect operations can be grouped into several categories.

Navigation functions support exploration within the Entrez databases:

- **esearch** performs a new Entrez search using terms in indexed fields.
- **elink** looks up neighbors (within a database) or links (between databases).

Entrez Programming Utilities Help

- **efilter** filters or restricts the results of a previous query.

Records can be retrieved in specified formats or as document summaries:

- **efetch** downloads records or reports in a designated format.

Desired fields from XML results can be extracted without writing a program:

- **xtract** converts XML into a table of data values.

Several additional functions are also provided:

- **einfo** obtains information on indexed fields in an Entrez database.
- **epost** uploads unique identifiers (UIDs) or sequence accession numbers.
- **nquire** sends a URL request to a web page or CGI service.

Entering Query Commands

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

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

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

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

Constructing Multi-Step Queries

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

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

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

Writing Commands on Multiple Lines

A query can be continued on the next line by typing the backslash ("\") UNIX escape character immediately before pressing the Return key. This makes query development and editing easier, allowing common query patterns to be saved in a text editor document and then copied and pasted into a terminal for execution. Continuing the query:

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

links to all protein sequences published in the neighbor articles.

Retrieving PubMed Reports

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

Entrez Programming Utilities Help

```
esearch -db pubmed -query "capsaicin cancer pain control" | \
efetch -format abstract
```

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

```
...
8. Curr Top Med Chem. 2011;11(17):2171-9.

The vanilloid agonist resiniferatoxin for interventional-based pain control.

Iadarola MJ, Mannes AJ.

Author information:
Neurobiology and Pain Therapeutics Section, Laboratory of Sensory Biology,
National Institute of Dental and Craniofacial Research, National Institutes
of Health, Bethesda MD 20892, USA.

The idea of selectively targeting nociceptive transmission at the level of
the peripheral nervous system is attractive from multiple perspectives,
particularly the potential lack of non-specific (non-targeted) CNS side
effects. Out of the multiple TRP channels involved in nociception, TRPV1
...
```

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

```
...
PMID- 21671877
OWN - NLM
STAT- MEDLINE
DA - 20110907
DCOM- 20120515
LR - 20131121
IS - 1873-4294 (Electronic)
IS - 1568-0266 (Linking)
VI - 11
IP - 17
DP - 2011
TI - The vanilloid agonist resiniferatoxin for interventional-based pain
control.
PG - 2171-9
AB - The idea of selectively targeting nociceptive transmission at the
level of the peripheral nervous system is attractive from multiple
...
```

Retrieving Sequence Reports

Nucleotide and protein records can be downloaded in FASTA format:

```
esearch -db protein -query "lycopene cyclase" | \
efetch -format fasta | \
grep '.'
```

Entrez Programming Utilities Help

which consists of a definition line followed by the sequence:

```
...
>gi|735882|gb|AAA81880.1| lycopene cyclase [Arabidopsis thaliana]
MDTLLKTPNKLDFFIPQFHGFERLCSNNPYPSRVRLGVKKRAIKIVSSVVSGSAALLDLVPETKKENLDF
ELPLYDTSKSQVVDLAIVGGGPAGLAVAQQVSEAGLSVCSIDPSPKLIWPNNYGVWVDEFEAMDLLDCLD
TTWSGAVVYVDEGVKKDLSRPYGRVNRKQLKSKMLQKCITNGVKFHQSKVTNVVHEEANSTVVCSDGVKI
QASVVLDATGFSRCLVQYDKPYNPGYQVAYGIIAEVDGHPFDVDKMVFMDWRDKHLDSYPELKERNSKIP
TFLYAMPFSSNRIFLEETSLVARPGLRMEDIQERMAARLKHLGINVKRIEEDERCVIPMGGPLPVLPQRV
VGIGGTAGMVHPSTGYMVARTLAAAPIVANAIVRYLGSPSSNSLRGDQLSAEVWRDLWPIERRRQREFFC
FGMDILLKLDLDATRRFFDAFFDLQPHYWHGFLSSRLFLPELLVFGLSLFSHASNTSRLEIMTKGTVPLA
KMINNLVQDRD
...
```

(The UNIX "grep" command at the end removes blank lines between FASTA records.)

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

```
...
LOCUS AAA81880 501 aa linear PLN 28-NOV-1995
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; rosids; malvids;
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)
JOURNAL Plant Physiol. 108 (3), 1343 (1995)
...
FEATURES Location/Qualifiers
source 1..501
/organism="Arabidopsis thaliana"
/db_xref="taxon:3702"
Protein 1..501
/product="lycopene cyclase"
transit_peptide 1..80
mat_peptide 81..501
/product="lycopene cyclase"
CDS 1..501
/gene="LYC"
/coded_by="L40176.1:2..1507"
ORIGIN
1 mdtllktpnk ldffipqfhg ferlcsnnpy psrvrlgvkk raikivssvv sgsaalldlv
61 petkkenldf elplydtsks qvvdlaivgg gpaglavaqq vseaglsvcs idpspkliwp
```

Entrez Programming Utilities Help

```
121 nnygvwvdef eamdlldcld ttwsgavvyv degvkkdlsr pygrvnrkql kskmlqkcit
181 ngvkfhqskv tnvvheeans tvvcsdgvki qasvvldatg fsrclvqydk pynpgyqvay
241 giiaevdghp fdvdkmvfmd wrdkhldsyp elkernskip tflyampfss nrifleetsl
301 varpglrmed iqermaarlk hlginvkrie edercvipmg gplpvlpqrv vgiggtagmv
361 hpstgymvar tlaaapivan aivrylgsps snslrgdqls aevwrdlwpi errrqreffc
421 fgmdillkld ldatrrffda ffdlqphywh gflssrlflp ellvfglslf shasntsrle
481 imtkgtvpla kminnlvqdr d
//
...
```

These formats are suitable for input into many popular sequence analysis programs.

Searching and Filtering

Restricting Query Results

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

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

Results can also be filtered by time with the -days argument. For example, use:

```
efilter -days 60 -datetype PDAT
```

to restrict results to the previous two months.

Qualifying Queries by Indexed Field

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

Commonly-used fields for PubMed queries include:

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

and a qualified query looks like:

```
"Tager H [AUTH] AND glucagon [TIAB]"
```

Filters that limit search results to subsets of PubMed include:

```
humans [MESH] historical article [FILT]
pharmacokinetics [MESH] loprovflybase [FILT]
chemically induced [SUBH] randomized controlled trial [FILT]
```

Entrez Programming Utilities Help

```
all child [FILT] clinical trial, phase ii [PTYP]
free full text [FILT] review [PTYP]
```

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

```
[ACCN] Accession [GENE] Gene Name [PROT] Protein Name
[ALL] All Fields [JOUR] Journal [SQID] SeqID String
[AUTH] Author [KYWD] Keyword [SLEN] Sequence Length
[GPRJ] BioProject [MLWT] Molecular Weight [SUBS] Substance Name
[ECNO] EC/RN Number [ORGN] Organism [WORD] Text Word
[FKEY] Feature Key [PACC] Primary Accession [TITL] Title
[FILT] Filter [PROP] Properties [UID] UID
```

and a sample query in the protein database is:

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

Additional examples of subset filters in sequence databases are:

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

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

A scripting example later in this document will produce a report giving the entire set of indexed fields for every Entrez database.

Examining Intermediate Results

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

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

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

produces:

Entrez Programming Utilities Help

```
<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 the elink function 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. Subsequent esearch commands can take a -db argument to override the database piped in from the previous step. (Piping the queries together is necessary for sharing the same web environment.) For example, the query:

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

uses truncation searching to return titles of papers with links to amyloid protein sequence and apolipoprotein gene records:

```
23962925 Genome analysis reveals insights into physiology and longevity ...
23959870 Low levels of copper disrupt brain amyloid-β homeostasis by ...
23371554 Genomic diversity and evolution of the head crest in the rock ...
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
1433291
...
```

without the need to keep track of the internal QueryKey values.

Working with 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>e59293</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:

```
1. PLoS One. 2013;8(3):e59293 ...
```

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. For example:

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

Xtract provides control over data conversion with a divide-and-conquer strategy using separate arguments for path exploration, conditional processing, item selection, and report formatting. Exploration subdivides XML records by context, independently processing every instance of a specified object. Any such subset may also be filtered by content, requiring the presence (or absence) of a particular data value in order to continue. Within the current XML subregion, selection finds every occurrence of each indicated element, printing their values as items are encountered. 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.

Entrez Programming Utilities Help

Entrez Programming Utilities Help

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

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

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

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

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

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

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

These commands are issued immediately after an exploration argument.

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

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

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

and apply to subsequent -element statements.

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

XML Document Summaries

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

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

will generate an XML document summary set:

```
<eSummaryResult>
<DocumentSummarySet status="OK">
<DbBuild>Build140105-1408.1</DbBuild>
<DocumentSummary>
<Id>19650888</Id>
<PubDate>2009 Aug 3</PubDate>
```

Entrez Programming Utilities Help

```
<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>
...
```

Piping the document summary output to:

```
xtract -outline
```

will give an indented overview of the XML structure hierarchy:

```
DocumentSummarySet
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:

```
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 ...
19578015 Bujakowska K Study of gene-targeted mouse models of splicing ...
19262028 Suto J Metabolic consequence of congenital asplenia ...
19248821 Fukumoto N Hypoalgesic behaviors of P/Q-type voltage-gated ...
18822497 Trishin AV [Protective activity of secreted proteins of ....
...
```

Entrez Programming Utilities Help

Processing Results with UNIX Utilities

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

```
esearch -db pubmed -query "Garber ED [AUTH] AND PNAS [JOUR]" | \
elink -related | \
efilter -query "mouse" | \
efetch -format docsum | \
xtract -pattern DocumentSummary -element Id SortFirstAuthor Title | \
sort -t $'\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 identified in ...
19578015 Bujakowska K Study of gene-targeted mouse models of splicing ...
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 pneumoniae ...
17142396 Fresno S A second galacturonic acid transferase is ...
16735743 Fresno S The ionic interaction of Klebsiella pneumoniae ...
...
```

Rather than always having to retype a series of common post-processing instructions, frequently used combinations of UNIX commands can be placed in a script, stored in an alias file (e.g., the user's .bash_profile), and executed by name. For example:

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

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

Titles can be passed to a pair of these UNIX alias commands:

```
esearch -db pubmed -query "Casadaban transposition immunity" | \
elink -related | \
efetch -format docsum | \
xtract -pattern DocumentSummary -element Title | \
word-at-a-time | \
sort-uniq-count-rank
```

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

Entrez Programming Utilities Help

```
299 of
178 the
114 transposition
100 and
94 in
93 mu
82 a
61 dna
61 tn3
56 transposon
50 bacteriophage
...
```

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:

```
2652 OPN1MW CBD|GCP|GOP|CBBM|COD5|OPN1MW1
5956 OPN1LW CBP|RCP|ROP|CBBM|COD5
```

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

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

Pubmed Article XML Records

The PubmedArticle object has a more detailed structure than the DocumentSummary, and is available for records in the pubmed database:

```
esearch -db pubmed -query "tetrachromacy" | \
efetch -format xml | \
xtract -outline
```

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

```
PubmedArticleSet
PubmedArticle
MedlineCitation
PMID
DateCreated
```

Entrez Programming Utilities Help

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

Using this information to craft a new xtract statement:

```
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:

```
23393278 Sabbah Troje Gray Hawryshyn
20884587 Jordan Deeb Bosten Mollon
18230593 Koshitaka Kinoshita Vorobyev Arikawa
17685813 Wachtler Doi Lee Sejnowski
```

```
16086150 Goldsmith Butler
...
```

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

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

Exploration of XML Sets

Individual PubmedArticle objects can be retrieved directly by efetch:

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

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

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

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

```
efetch -db pubmed -id 6092233,6301692,781293 -format xml | \
xtract -pattern PubmedArticle -element MedlineCitation/PMID \
-element Initials LastName
```

will print all author initials and then all author last names:

```
6092233 IL CR RK Calderon Contopoulou Mortimer
6301692 MA NR Krasnow Cozzarelli
781293 MJ Casadaban
```

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

Entrez Programming Utilities Help

Entrez Programming Utilities Help

```
efetch -db pubmed -id 6092233,6301692,781293 -format xml | \
xtract -pattern PubmedArticle -element MedlineCitation/PMID \
-block Author -element Initials LastName
```

which restores the correct association of initials and last name:

```
6092233 IL Calderon CR Contopoulou RK Mortimer
6301692 MA Krasnow NR Cozzarelli
781293 MJ Casadaban
```

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

```
efetch -db pubmed -id 6092233,6301692,781293 -format xml | \
xtract -pattern PubmedArticle -element MedlineCitation/PMID \
-block Author -sep " " -element Initials,LastName
```

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

```
6092233 IL Calderon CR Contopoulou RK Mortimer
6301692 MA Krasnow NR Cozzarelli
781293 MJ Casadaban
```

Recording Values in Variables

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

```
efetch -db pubmed -id 6092233,6301692,781293 -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:

```
6092233 IL Calderon
6092233 CR Contopoulou
6092233 RK Mortimer
6301692 MA Krasnow
6301692 NR Cozzarelli
781293 MJ Casadaban
```

Variables can be initialized with a literal value in parentheses:

```
efetch -db pubmed -id 6092233,6301692,781293 -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:

```
6092233 IL Calderon, CR Contopoulou, RK Mortimer
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 6092233,6301692,781293 -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 names, counts the number of authors present, and prints the date each record was published and the date it was entered into PubMed, sorting the results by the computed author count:

```
781293 Casadaban 1 1976 Jul 1976-10-02
6301692 Krasnow/Cozzarelli 2 1983 Apr 1983-06-17
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>1976</Year>
<Month>Jul</Month>
<Day>5</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>
```

Entrez Programming Utilities Help

```
<DescriptorName>Recombination, Genetic</DescriptorName>
<QualifierName>genetics</QualifierName>
</MeshHeading>
<MeshHeading>
<DescriptorName>Saccharomyces cerevisiae</DescriptorName>
<QualifierName>genetics</QualifierName>
<QualifierName>radiation effects</QualifierName>
</MeshHeading>
<MeshHeading>
<DescriptorName>Saccharomyces cerevisiae Proteins</DescriptorName>
</MeshHeading>
</MeshHeadingList>
...
```

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 1937004 -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:

```
1937004
Adenosine Triphosphatases
Amino Acid Sequence
Base Sequence
...
Recombination, Genetic/genetics
Saccharomyces cerevisiae/genetics/radiation effects
Saccharomyces cerevisiae Proteins
```

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

```
...
<MeshHeading>
<DescriptorName MajorTopicYN="N">Saccharomyces cerevisiae</DescriptorName>
<QualifierName MajorTopicYN="Y">genetics</QualifierName>
<QualifierName MajorTopicYN="N">radiation effects</QualifierName>
</MeshHeading>
...
```

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

```
efetch -db pubmed -id 1937004 -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"
```

Entrez Programming Utilities Help

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

```
1937004
|N|Adenosine Triphosphatases
|N|Amino Acid Sequence
|N|Base Sequence
...
|N|Recombination, Genetic/|N|genetics
|N|Saccharomyces cerevisiae/|Y|genetics/|N|radiation effects
|Y|Saccharomyces cerevisiae Proteins
```

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:

```
1937004
Adenosine Triphosphatases
Amino Acid Sequence
Base Sequence
...
Recombination, Genetic/genetics
Saccharomyces cerevisiae/*genetics/radiation effects
*Saccharomyces cerevisiae Proteins
```

Conditional Processing

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

```
efetch -db pubmed -id 1937004 -format xml | \
xtract -pattern PubmedArticle -PMID MedlineCitation/PMID \
-group MeshHeading \
-block MeshHeading -match QualifierName \
-subset DescriptorName -TERM "DescriptorName" -MAJR "@MajorTopicYN" \
-subset QualifierName -tab "\n" \
-element "&PMID","&TERM","&MAJR",QualifierName,"@MajorTopicYN" \
-block MeshHeading -avoid QualifierName \
-subset DescriptorName -tab "\n" \
-element "&PMID",DescriptorName,"@MajorTopicYN"
```

has separate sections for MeSH terms with and without subheadings, and produces results that are suitable for importing into a database or spreadsheet program:

```
1937004 Adenosine Triphosphatases N
1937004 Amino Acid Sequence N
1937004 Base Sequence N
...
1937004 Recombination, Genetic N genetics N
1937004 Saccharomyces cerevisiae N genetics Y
1937004 Saccharomyces cerevisiae N radiation effects N
1937004 Saccharomyces cerevisiae Proteins Y
```

Entrez Programming Utilities Help

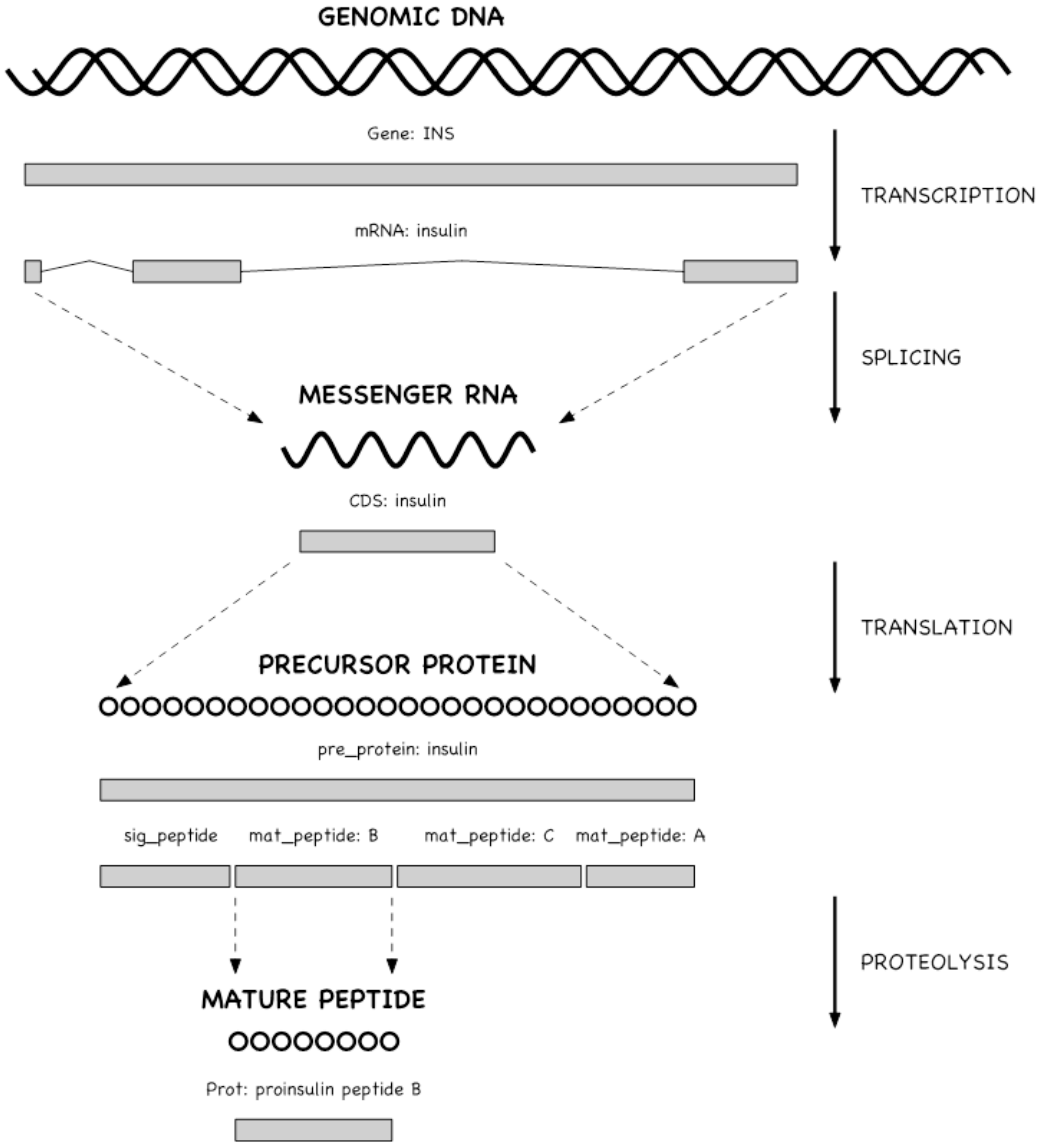
Multiple -match or -avoid conditions are specified with -and and -or commands.

Processing 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 location on genomic DNA corresponds to 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 sequence.



Entrez Programming Utilities Help

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 to display in a /peptide qualifier, even if a mature peptide is not instantiated.

Sequence Records in INSDSeq XML

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

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

returns a set of INSDSeq objects:

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

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

```
FEATURES Location/Qualifiers
source 1..17
/organism="Conus imperialis"
/db_xref="taxon:35631"
/country="Philippines"
Protein <1..17
/product="alpha-conotoxin ImI precursor"
```

Entrez Programming Utilities Help

```
mat_peptide 5..16
/product="alpha-conotoxin ImI"
/note="the C-terminal glycine of the precursor is post
translationally removed"
/calculated_mol_wt=1357
/peptide="GCCSDPRCAWRC"
CDS 1..17
/coded_by="AY159318.1:<1..54"
/note="nAChR antagonist"
```

in a structured feature table:

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

Feature and qualifier names are indicated in data values, not XML element tags, and require -match to select the desired object and content.

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

Entrez Programming Utilities Help

instructions. 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
```

in an isolated command prints a new xtract statement that will produce a table of qualifier values from mature peptide features with complete locations. That statement can then be copied and pasted into other queries.

Incorporating the same xtract 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 the accession number, calculated peptide length, product name, and peptide sequence:

```
AGO59814.1 32 del13b conotoxin DCPTSCPTTCANGWECCKGYPCVRQHCSGCNH
AAO33169.1 16 alpha-conotoxin GIC GCCSHPACAGNNQHIC
ADB65788.1 20 conotoxin Cal 16 LEMQGCVCNANAKFCCGEGR
AAN86573.1 30 kappaA-conotoxin SIVA QKSLVPSVITTCCGYDPGTMCPPCRCTNSC
AAF23167.1 31 BeTX toxin CRAEGTYCENDSQCCLNECCWGGCGHPCRHP
ADB65789.1 20 conotoxin Cal 16 LEMQGCVCNANAKFCCGEGR
AAD31912.1 21 alpha A conotoxin Tx1 PECCSDPRCNSSHPELCGGRR
ABW16858.1 15 marmophin DWEYHAHPKPNSFWT
...
```

Piping the results to a series of UNIX commands:

```
grep -i conotoxin | \
awk -F '\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:

```
AAN78128.1 alpha-conotoxin ImI GCCSDPRCAWRC
AAN78127.1 alpha-conotoxin ImII ACCSDRRCRWRC
ADB43130.1 conotoxin Cal 1a KCCKRHHGCHPCGRK
ADB43131.1 conotoxin Cal 1b LCCKRHHGCHPCGRT
AAO33169.1 alpha-conotoxin GIC GCCSHPACAGNNQHIC
ADB43128.1 conotoxin Cal 5.1 DPAPCCQHPIETCCRR
AAD31913.1 alpha A conotoxin Tx2 PECCSHPACNVDHPEICR
ADB43129.1 conotoxin Cal 5.2 MIQRSQCCAVKKNCCHVG
ADD97803.1 conotoxin Cal 1.2 AGCCPTIMYKTGACRTNRCR
```

Entrez Programming Utilities Help

```
ADB65789.1 conotoxin Cal 16 LEMQGCVCNANAKFCCGEGR
AAD31912.1 alpha A conotoxin Tx1 PECCSDPRCNSSHPELCGGRR
AAN78279.1 conotoxin Vx-II WIDPSHYCCCGGGCTDDCVNC
ADB43125.1 conotoxin Cal 14.2 GCPADCPNTCDSSNKCSPGFPG
ADD97802.1 conotoxin Cal 6.4 GCWLCLGPNACCRGSVCHDYCPR
CAH64846.1 four-loop conotoxin CRPSGSPCGVTSICCGRCSRGKCT
AAD31915.1 O-superfamily conotoxin TxO2 CYDSGTSCNTGNQCCSGWCIFVCL
AAD31916.1 O-superfamily conotoxin TxO3 CYDGGTSCDSGIQCCSGWCIFVCF
AAD31920.1 omega conotoxin SVIA mutant 1 CRPSGSPCGVTSICCGRCYRGKCT
AAD31921.1 omega conotoxin SVIA mutant 2 CRPSGSPCGVTSICCGRCSRGKCT
...
```

Extensive Modification of Tabular Output

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

```
efetch -db nuccore -id "GQ370762.1" -format gbc | \
xtract -pattern INSDSeq \
-pfx ">Feature " -tab "" -first INSDSeqid \
-group INSDFeature \
-avoid "INSDFeature_key:source" \
-FKEY INSDFeature_key \
-block INSDInterval \
-pfx "\n" -tab "" \
-element INSDInterval_from,INSDInterval_to \
INSDInterval_point,INSDInterval_point \
-pfx "\t" -tab "" \
-element "&FKEY" -FKEY "" \
-block INSDQualifier \
-avoid "INSDQualifier_name:transcription" \
-and "INSDQualifier_name:translation" \
-and "INSDQualifier_name:peptide" \
-pfx "\n\t\t\t" -tab "" \
-element INSDQualifier_name,INSDQualifier_value
```

clears the feature key variable after its first use to reproduce the 5-column feature table format used for GenBank submissions:

```
>Feature gb|GQ370762.1|
51 1474 gene
gene HBB
51 142 mRNA
273 495
1346 1474
gene HBB
product beta-globin
51 142 CDS
273 495
1346 1474
gene HBB
codon_start 1
transl_table 1
```

Entrez Programming Utilities Help

```
product beta-globin
protein_id ACU56984.1
db_xref GI:256028940
```

(A location interval will have either an INSDInterval_from and INSDInterval_to pair or a single INSDInterval_point. Additional work would be required to support the 5' and 3' partial flags for features with incomplete locations.)

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

Advanced Topics

Storing Common Phrases in Alias Files

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

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

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

the esearch line in:

```
esearch -alias query_aliases -db nlmcatalog -query "Science (#jour_filt)" |
\
efetch -format docsum | \
xtract -pattern DocumentSummary -element ISOAbbreviation \
-subset ISSNInfo -sep "|" -element issn,issntype
```

will be expanded to:

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

with the query producing:

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

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

```
eproxy -alias query_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.

Sending Results to Scripts or Spreadsheets

Specific fields can be extracted from XML results and passed to other programs or scripts for further computational analysis. For example:

Entrez Programming Utilities Help

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

obtains a series of homologous genes and writes a tab-delimited data stream, converting the original 0-based gene coordinates to 1-based positions suitable for retrieving sequence regions:

```
1276381 NT_078265.2 19282704 19277265
741097 NC_006479.3 103229228 103150285
698696 NC_007868.1 104081544 104008955
...
```

(The use of -block limits the subsequent -element search to fields within the GenomicInfoType block, and prevents it from capturing an additional ChrStart item that occurs elsewhere in the DocumentSummary XML.)

Given a shell script named "upstream.sh":

```
#!/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
```

the data lines can be piped through:

Entrez Programming Utilities Help

```
upstream.sh 500
```

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

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

Tab-delimited data can also be saved directly to a file with the right angle bracket (">") UNIX output redirection character. The resulting file is suitable for importing into a database or spreadsheet program.

Advanced ESearch and ELink Commands

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

```
esearch -db pubmed -query "opsin gene conversion" -sort "last author" | \
efetch -format docsum | \
xtract -pattern DocumentSummary -element Id 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. For example, the query:

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

produces a ranked list of the ten most cited authors:

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

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
```

Entrez Programming Utilities Help

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 "free full text [FILT]" | \
efetch -format uid | \
xargs -n 1 elink -db pubmed -cmd prlinks -mode ref -http get -id
```

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

Xtract Special Topics

Self-closing tags of the standard form:

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

or alternative form:

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

can be indicated by appending parentheses after the tag:

```
-element "Na-strand()"
```

and the tag name will be printed when found. If a string is placed in the parentheses, it will be printed instead of the name. If the tag contains an attribute:

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

the -present filter can be used to print a particular string based on the value:

```
-block Seq-interval_strand -present plus -element "Na-strand(+)" \
-block Seq-interval_strand -present minus -element "Na-strand(-)"
```

The -lbl argument can be used to print arbitrary text:

```
-group MeshHeading \
-block MeshHeading -match QualifierName -lbl "subheadings present" \
-block MeshHeading -avoid QualifierName -lbl "subheadings missing"
```

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

Entrez Programming Utilities Help

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 an extraction in pseudo code can clarify relative nesting levels when building complex commands. The extraction command:

```
xtract -pattern PubmedArticle \
-block Author -element Initials,LastName \
-block MeshHeading -element DescriptorName \
-subset QualifierName -element QualifierName
```

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

```
for each PubmedArticle {
for each Author {
print Initials LastName
}
for each MeshHeading {
print DescriptorName
for each QualifierName {
print QualifierName
}
}
}
```

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

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

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

Using -synopsis instead of -outline produces a table of unique path counts:

```
1 DocumentSummary
1 DocumentSummary/ArticleIds
5 DocumentSummary/ArticleIds/ArticleId
5 DocumentSummary/ArticleIds/ArticleId/IdType
5 DocumentSummary/ArticleIds/ArticleId/IdTypeN
5 DocumentSummary/ArticleIds/ArticleId/Value
1 DocumentSummary/Attributes
```

```
1 DocumentSummary/Authors
2 DocumentSummary/Authors/Author
2 DocumentSummary/Authors/Author/AuthType
2 DocumentSummary/Authors/Author/ClusterID
2 DocumentSummary/Authors/Author/Name
...
```

Heterogeneous Data

XML objects can contain a heterogeneous mix of components. For example:

```
efetch -db pubmed -id 21433338,17247418 -format xml
```

returns a mixture of book and journal records:

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

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

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

separately prints the entirety of each XML component:

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

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

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

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

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

Entrez Programming Utilities Help

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 author lists could also be selected by using -present with 'Type="authors"' or 'Type="editors"' attribute text values, but exploring by "Parent/Child" is a general position-based solution to the problem.)

Recursive Definitions

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

Retrieving a set of taxonomy records:

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

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

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

Entrez Programming Utilities Help

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

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

```
-Division
-Group
-Branch
-Block
-Section
-Subset
-Unit
```

Extraction of data with a capitalized -Group argument:

```
efetch -db taxonomy -id 9606,7227 -format xml | \
xtract -pattern TaxaSet -Group Taxon -sfx "\n" -tab "" -ret "" \
-first TaxId,ScientificName,GenbankCommonName
```

behaves as desired and returns information for the main entries:

```
9606 Homo sapiens human
7227 Drosophila melanogaster fruit fly
```

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

```
9606 Homo sapiens human
2759 Eukaryota
33154 Opisthokonta
33208 Metazoa
...
```

Similarly, use of a capitalized -Block argument:

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

correctly finds all appropriate accessions in Entrezgene records:

```
CP002684 AEE27818 DQ446230 ABE65601
CP002684 AEE27823 CP002684 AEE27824 CP002684 AEE27825
```

(The alternative search algorithm is significantly slower than the regular expression version in the current implementation, so capitalized exploration arguments should only be used when necessary to handle recursive elements.)

Entrez Programming Utilities Help

(There is no capitalized version of the top-level -pattern organizer.)

The -trim argument will skip past the first tag (e.g., <Taxon>) so that subsequent explorations are able to visit the internal objects:

```
efetch -db taxonomy -id 9606,7227 -format xml | \
xtract -pattern TaxaSet -Group Taxon -trim -block Taxon -sfx "\n" \
-tab "" -ret "" -first TaxId,ScientificName,GenbankCommonName
```

That command returns:

```
131567 cellular organisms
2759 Eukaryota
33154 Opisthokonta
33208 Metazoa
...
```

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:

```
nquire -url "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi" \
-db pubmed -term insulin
```

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

```
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?
db=pubmed&term=insulin
```

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

```
-url http://eutils.ncbi.nlm.nih.gov entrez/eutils efetch.fcgi
```

is converted to:

```
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi
```

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. For example:

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

will be sent as:

Entrez Programming Utilities Help

```
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:

```
ncbi_url http://www.ncbi.nlm.nih.gov
eutils_url http://eutils.ncbi.nlm.nih.gov/entrez/eutils
```

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:

```
xmllint --format -
```

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

For example, executing a script file containing:

```
#!/bin/bash -norc

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

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

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

Entrez Programming Utilities Help

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

sleep 1
done
```

will obtain the list of Entrez databases, and then return the abbreviations, names, and descriptions of indexed search fields, for each individual database:

```
...
epigenomics

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

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.

Complex Behavior Without Scripting

Writing a script to process every database or UID can sometimes be avoided by creative use of the UNIX xargs and sh commands. Within the "sh" command string, the database argument passed by xargs is substituted at the "$0" variable. All of the sh commands are run separately on each database:

```
einfo -dbs | xtract -pattern DbName -element DbName | \
sort | xargs -n 1 sh -c 'einfo -db $0 | \
xtract -pattern DbInfo -tab "\n" -element DbName \
-block Field -pfx "\n[" -sfx "]" -element Name \
-pfx "" -sfx "" -element FullName | \
sort -k 2 | expand'
```

This produces results that are alphabetized for each database:

Entrez Programming Utilities Help

```
...
epigenomics
[ACCN] Accession
[ALL] All Fields
[AUTH] Author
[CDAT] Create Date
[DOCT] Document Type
[FILT] Filter
...
```

Examples

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

If the sort-uniq-count-rank alias, used in some of these examples and presented earlier in this document, is not already in the user's .bash_profile configuration file, it can be replaced by the following commands:

```
sort -f | uniq -i -c | \
perl -pe 's/\s*(\d+)\s(.+)/$1\t$2/' | \
sort -t $'\t' -k 1,1nr -k 2f
```

Author Frequency

Who are the most prolific authors on rattlesnake phospholipase?

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

This search produces:

```
73 Gutiérrez JM
73 Lomonte B
45 Marangoni S
45 Soares AM
42 Giglio JR
39 Bon C
...
```

Protein Homolog

Is there a mammalian equivalent of lycopene cyclase?

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

Entrez Programming Utilities Help

In the resulting list of GenBank division codes:

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

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

Longest Sequences

What are the longest known insulin precursor sequences?

```
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
```

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?

```
esearch -db structure -query "crotalus [ORGN] AND phospholipase A2" | \
elink -related | \
```

Entrez Programming Utilities Help

```
efilter -query "archaea [ORGN]" | \
efetch -format docsum | \
xtract -pattern DocumentSummary \
-match "PdbClass:Hydrolase" \
-element PdbDescr | \
sort -f | uniq -i
```

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

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

Taxonomy Search

Which organisms contain an annotated RefSeq genome MatK gene?

```
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
Acorus calamus
Adiantum capillus-veneris
...
```

Exon Counts

How many exons are in each dystrophin transcript variant?

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

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

```
NC_000023.11 33339608 31119221
```

The values are adjusted by an awk command:

Entrez Programming Utilities Help

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

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

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

These are then passed as arguments to efetch:

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

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

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

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

Final processing sorts by exon number:

```
grep -i dystrophin | sort -k 2,2nr -k 4,4nr
```

resulting in a table of exon counts and transcript lengths:

```
NC_000023.11 79 NM_004010.3 14083 dystrophin, transcript variant Dp427p2
NC_000023.11 79 NM_000109.3 14069 dystrophin, transcript variant Dp427c
NC_000023.11 79 NM_004009.3 14000 dystrophin, transcript variant Dp427p1
NC_000023.11 79 NM_004006.2 13993 dystrophin, transcript variant Dp427m
NC_000023.11 78 XM_006724468.1 13920 dystrophin, transcript variant X1
NC_000023.11 78 XM_006724469.1 13802 dystrophin, transcript variant X2
NC_000023.11 77 XM_006724470.1 13881 dystrophin, transcript variant X3
...
```

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

Entrez Programming Utilities Help

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

```
IL3RA NC_000024.10 1336601 1382689 1
LOC102724575 NC_000024.10 1365567 1368017 1
LOC101928032 NC_000024.10 1378130 1382392 2
SLC25A6 NC_000024.10 1386152 1392146 2
ASMTL-AS1 NC_000024.10 1400531 1415421 1
ASMTL NC_000024.10 1403139 1453794 2
P2RY8 NC_000024.10 1462572 1537144 2
AKAP17A NC_000024.10 1591593 1602520 1
ASMT NC_000024.10 1595455 1643081 1
```

(The "ChrLoc:Y" match is necessary because certain genes are present on both the X and Y chromosomes.)

Amino Acid Substitutions

What are the missense products of green-sensitive opsin?

```
esearch -db gene -query "CBD [GENE] AND human [ORGN]" | \
elink -target snp | \
efetch -format xml | \
xtract -pattern Rs -RSID Rs@rsId \
-block FxnSet -match @fxnClass:missense \
-sep "." -element "&RSID" @protAcc,@protVer @aaPosition \
-tab "\n" -element @residue | \
sort -t $'\t' -k 2,2 -k 3,3n -k 4,4 | uniq
```

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

```
104894915 NP_000504.1 93 K
200470120 NP_000504.1 110 I
201569525 NP_000504.1 115 S
372044027 NP_000504.1 173 V
267606927 NP_000504.1 176 R
...
```

The results can be piped to a shell script:

```
#!/bin/bash -norc

seq=""
last=""

while read rsid accn pos res; do
if [ "$accn" != "$last" ]; then
insd=`efetch -db protein -id "$accn" -format gbc < /dev/null`
seq=`echo $insd | xtract -pattern INSDSeq -element INSDSeq_sequence`
last=$accn
fi
```

Entrez Programming Utilities Help

```
echo ">rs$rsid [$accn $res@$pos]"
if [ $pos -gt 1 ]; then
echo ${seq:0:$pos-1} | fold -w 50
fi
echo $res
if [ $pos -lt ${#seq} ]; then
echo ${seq:$pos} | fold -w 50
fi
done
```

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

```
>rs104894915 [NP_000504.1 K@93]
maqqwslqrlagrhpqdsyedstqssiftytnsnstrgpfegpnyhiapr
wvyhltsvwmifvviasvftnglvlaatmkfkklrhplnwil
K
nlavadlaetviastisvvnqvygyfvlghpmcvlegytvslcgitglws
laiiswerwmvvckpfgnvrfdaklaivgiafswiwaavwtappifgwsr
ywphglktscgpdvfsgssypgvqsymivlmvtccitplsiivlcylqvw
...
```

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 -match \"LastName:$auth\"" \
"-sep \" \" -element LastName,Initials"`
done

query=`echo "$query | sort -t \$'\\t' -k 2f -k 1,1n | xtract -fuse pubmed"`

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

Entrez Programming Utilities Help

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 > author_patterns.htm
```

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

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

The sorted lines are passed to the (experimental) xtract -fuse function, which combines them into PubMed query URLs, one for each author pattern:

```
http://www.ncbi.nlm.nih.gov/pubmed/1827084,2954879,3020001,...
```

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?

Entrez Programming Utilities Help

```
einfo -db pubmed | \
xtract -pattern Field \
-match "IsDate:Y" -and "IsHidden:N" \
-pfx "[" -sfx "]" -element Name \
-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
```

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:

```
econtact -email author@institution.edu -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:

Navigation functions include esearch, elink, and efilter:

```
esearch

-db Database name
-query Query string

-sort Result presentation order

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

-label Alias for query step

elink

-related Neighbors in same database
```

Entrez Programming Utilities Help

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

-db Database name
-id Unique identifier(s)

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

-batch Bypass Entrez history mechanism

-label Alias for query step

efilter

-query Query string

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

-label Alias for query step
```

The record retrieval function is efetch:

```
efetch

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

-db Database name
-id Unique identifier or accession number

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

The xtract function is used for processing XML data:

```
xtract

Exploration Argument Hierarchy

-pattern (Highest Rank)
-division
-group
-branch
-block
-section
```

Entrez Programming Utilities Help

```
-subset
-unit (Lowest Rank)

Conditional Execution

-match Element [:value] required
-avoid Skips if element matches
-present Requires indicated data pattern
-absent Pattern must not be present
-and All tests must pass
-or Any passing test suffices
-position Must be at given location in list

Format Customization

-ret Overrides line break between patterns
-tab Replaces tab character between fields
-sep Separator between group members
-pfx Prefix to print before group
-sfx Suffix to print after group

Item Selection

-element Print all items that match tag name
-first Only prints value of first item
-last Only prints value of last item

Miscellaneous Arguments

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

-insd Generate INSDSeq extraction commands

-lbl Inserts arbitrary text

-element Construct Examples

Tag Caption
Group Initials,LastName
Parent/Child MedlineCitation/PMID
Attribute DescriptorName@MajorTopicYN
Object Count "#Author"
Item Length "%Title"
Variable "&ACCN"
```

Several additional functions are provided by EDirect:

```
einfo

-db Database name
```

Entrez Programming Utilities Help

```
-dbs Get all database names

epost

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

eproxy

-alias File of aliases
-pipe Read aliases from stdin

econtact

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

nquire

-url Base URL for external search
-http "get" uses HTTP GET instead of POST
```

In addition, -email and -tool are available in all E-utilities-calling functions to override default values, -silent will suppress link failure retry messages, -verbose will print the <ENTREZ_DIRECT> field values at each step, -debug can be used to assist with debugging a script by printing the internal URL query and XML results of each step, -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, -base will specify a particular server for quality assurance testing, and -help will print the list of arguments for each function.

EFetch Formats

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

```
-db -format -mode Report Type
___ _______ _____ ___________

(all)
docsum DocumentSummarySet XML
docsum json DocumentSummarySet JSON
full Same as native except for mesh
uid Unique Identifier List
url Entrez URL
xml Converted to -format full -mode xml

bioproject
native BioProject Report
native xml RecordSet XML

biosample
native BioSample Report
```

Entrez Programming Utilities Help

```
native xml BioSampleSet XML

biosystems
native xml Sys-set XML

gds
native xml RecordSet XML
summary Summary

gene
gene_table Gene Table
native Gene Report
native asn.1 Entrezgene ASN.1
native xml Entrezgene-Set XML
tabular Tabular Report

homologene
alignmentscores Alignment Scores
fasta FASTA
homologene Homologene Report
native Homologene List
native asn.1 HG-Entry ASN.1
native xml Entrez-Homologene-Set XML

mesh
full Full Record
native MeSH Report
native xml RecordSet XML

nlmcatalog
native Full Record
native xml NLMCatalogRecordSet XML

pmc
medline MEDLINE
native xml pmc-articleset XML

pubmed
abstract Abstract
medline MEDLINE
native asn.1 Pubmed-entry ASN.1
native xml PubmedArticleSet XML

(sequences)
acc Accession Number
est EST Report
fasta FASTA
fasta xml TinySeq XML
fasta_cds_aa FASTA of CDS Products
fasta_cds_na FASTA of Coding Regions
ft Feature Table
gb GenBank Flatfile
```

Entrez Programming Utilities Help

```
gb xml GBSet XML
gbc xml INSDSet XML
gbwithparts GenBank with Contig Sequences
gp GenPept Flatfile
gp xml GBSet XML
gpc xml INSDSet XML
gss GSS Report
native text Seq-entry ASN.1
native xml Bioseq-set XML
seqid Seq-id ASN.1

snp
chr Chromosome Report
docset Summary
fasta FASTA
flt Flat File
native asn.1 Rs ASN.1
native xml ExchangeSet XML
rsr RS Cluster Report
ssexemplar SS Exemplar List

sra
native xml EXPERIMENT_PACKAGE_SET XML

structure
mmdb Ncbi-mime-asn1 strucseq ASN.1
native MMDB Report
native xml RecordSet XML

taxonomy
native Taxonomy List
native xml TaxaSet XML
```

ESearch Sort Order

ESearch -sort values for several databases are listed below:

```
-db -sort
___ _____

gene
Chromosome
Gene Weight
Name
Relevance

pubmed
First Author
Journal
Last Author
Pub Date
Recently Added
```

Entrez Programming Utilities Help

```
Relevance
Title

(sequences)
Accession
Date Modified
Date Released
Organism Name
Taxonomy ID
```

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:

```
grep Matches patterns using regular expressions

-i Ignore case
-v Invert search
-w Search expression as a word

-e Specify individual pattern

-c Only count number of matches
-n Print line numbers

sort Sorts lines of text

-f Ignore case
-n Numeric comparison
-r Reverse result order
```

Entrez Programming Utilities Help

```
-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
```

Modify contents:

```
sed Replaces text strings

-e Specify individual expression

tr Translates characters

-d Delete character
```

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
```

Entrez Programming Utilities Help

```
-s Suppress lines with no delimiters

head Prints first lines

-n Number of lines

tail Prints last lines

-n Number of lines
```

Miscellaneous:

```
wc Counts words, lines, or characters

-c Characters
-l Lines
-w Words

xargs Constructs arguments

-n Number of words per batch
```

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

```
cd Changes directory

/ Root
~ Home
. Current
.. Parent
- Previous

ls Lists file names

-1 One entry per line
-a Show files beginning with dot (.)
-l List in long format
-R Recursively explore subdirectories
```

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

Release Notes

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.

Entrez Programming Utilities Help

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.

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.