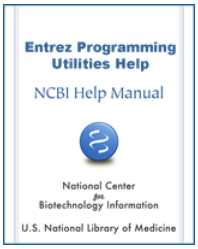
U.S. National Library of Medicine
National Center for Biotechnology Information

NLM Citation: Kans J. Entrez Direct Examples. 2013 Apr 23 [Updated 2024 Apr 12]. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-.
Bookshelf URL: https://www.ncbi.nlm.nih.gov/books/



Entrez Direct Examples

Jonathan Kans, PhD[1]

Created: April 23, 2013; Updated: April 12, 2024.

Some early EDirect examples required post-processing with shell script:

```
start=$(( start + 1 ))
stop=$(( stop + 1 ))
```

or awk commands:

```
awk -F '\t' -v 'OFS=\t' '{print $1, $2+1, $3+1}'
```

to increment 0-based sequence coordinates so that the proper region was retrieved:

```
efetch -db nuccore -id $accn -format fasta -seq_start $start -seq_stop $stop
```

This led to -element derivatives that could do simple adjustments within an xtract command:

```
-element ChrAccVer -inc ChrStart -1-based ChrStop
```

and to efetch arguments that could take 0-based coordinates directly:

```
efetch -db nuccore -id $accn -format fasta -chr_start $start -chr_stop $stop
```

These helped eliminate the need for scripts to perform otherwise trivial modifications on extracted data.

More recent work allows easier after-the-fact numeric manipulation using the filter-columns:

```
filter-columns '10 <= $2 && $2 <= 30'
```

and print-columns:

```
print-columns '$1, $2+1, $3-1, "\042" $4 "\042", tolower($5), log($3), total += $2'
```

scripts, which accept column designators in an argument and pass them to internal awk commands. The NF (number of fields) and NR (current record number) built-in variables can also be used, as can YR (for year) and DT (for date in YYYY-MM-DD format).

PubMed

Author Frequency

Who are the most prolific authors on rattlesnake phospholipase?

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

Corresponding author.

```
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
```

Placing author names on separate lines allows sort-uniq-count-rank to produce a frequency table:

```
87    Lomonte B
77    Gutiérrez JM
64    Soares AM
53    Marangoni S
43    Giglio JR
39    Bon C
...
```

Publication Distribution

When were the most papers about Legionnaires disease published?

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

In this case sort-uniq-count-rank reports the number of selected papers per year:

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

Using −year "PubDate/*" on a PubmedArticle record takes the first four-digit number it encounters, and the result does not need to be trimmed:

```
esearch -db pubmed -query "legionnaires disease [TITL]" |
efetch -format xml |
xtract -pattern PubmedArticle -year "PubDate/*" |
sort-uniq-count-rank
```

Treatment Locations

What is the geographic distribution of sepsis treatment studies?

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

This returns the number of articles ranked by country or region:

```
660    United States
250    Spain
182    Germany
```

```
168    India
155    Taiwan
139    Japan
126    China
124    France
123    Europe
...
```

Note that England and United Kingdom will appear as two separate entries.

Indexed Date Fields

What date fields are indexed for PubMed?

```
einfo -db pubmed |
xtract -pattern Field \
  -if IsDate -equals Y -and IsHidden -equals N \
    -pfx "[" -sfx "]" -element Name \
    -pfx "" -sfx "" -element FullName |
sort -k 2f | expand
```

Indexed dates are shown with field abbreviations and descriptive names:

```
[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?

```
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 \
      -tab "" -pfx '<p><a href="' -sfx '">' -doi ArticleId \
      -tab "\n" -pfx '' -sfx '</a></p>' -element "&PMID" |
transmute -format
```

The −doi command extracts the DOIs and constructs URL references. The −pfx and −sfx arguments used here enclose each PMID with a clickable link to its DOI:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html>
  <body>
    <p>
      <a href="https://doi.org/10.1038%2Fleu.2013.340">24496283</a>
    </p>
    <p>
      <a href="https://doi.org/10.1073%2Fpnas.1310656110">23818607</a>
    </p>
    ...
  </body>
</html>
```

Reference Formatting

How were references in the main EDirect documentation page formatted?

```
efetch -db pubmed -format docsum -id 26931183 31738401 31600197 \
  29718389 14728215 11449725 8743683 31114887 23175613 |
```

Relevant document summary fields were collected with:

```
xtract -set Set -rec Rec -pattern DocumentSummary \
  -sep ", " -sfx "." -NM Name -clr \
  -sfx "." -SRC Source -clr \
  -pfx ":" -PG Pages -clr \
  -pfx "(PMID " -sfx ".)" -ID Id -clr \
  -group ArticleId -if IdType -equals doi \
    -pfx "https://doi.org/" -sfx "." -DOI Value -PG "(.)" -clr \
  -group DocumentSummary \
    -wrp Sort -lower "&NM" \
    -wrp Name -element "&NM" \
    -wrp Title -element Title \
    -wrp Source -element "&SRC" \
    -wrp Year -year PubDate \
    -wrp Pages -element "&PG" \
    -wrp DOI -element "&DOI" \
    -wrp Id -element "&ID" |
```

This generated intermediate XML with partial formatting:

```
...
<Rec>
  <Sort>wei ch, allot a, leaman r, lu z.</Sort>
  <Name>Wei CH, Allot A, Leaman R, Lu Z.</Name>
  <Title>PubTator central: automated concept ... full text articles.</Title>
  <Source>Nucleic Acids Res.</Source>
  <Year>2019</Year>
  <Pages>.</Pages>
  <DOI>https://doi.org/10.1093/nar/gkz389.</DOI>
  <Id>(PMID 31114887.)</Id>
</Rec>
...
```

References were sorted by the first author's last name:

```
xtract -set Set -rec Rec -pattern Rec -sort Sort |
```

and then printed with:

```
xtract -pattern Rec -deq "\n\n" -tab " " -sep "" \
  -element Name Title Source Year,Pages DOI Id
```

followed by manual editing of hyphenated initials:

```
...
Wei C-H, Allot A, Leaman R, Lu Z. PubTator central: automated concept
annotation for biomedical full text articles. Nucleic Acids Res. 2019.
https://doi.org/10.1093/nar/gkz389. (PMID 31114887.)
...
```

Using xtract **–reg** and **–exp** regular expressions to guide xtract **–replace**:

```
efetch -db pubmed -id 31114887 -format xml |
xtract -pattern PubmedArticle -block Author -deq "\n" \
  -element LastName -reg "[a-z .]" -exp "" -replace ForeName
```

can generate hyphenated initials directly from the PubmedArticle ForeName field:

```
<Author>
  <LastName>Wei</LastName>
  <ForeName>Chih-Hsuan</ForeName>
  <Initials>CH</Initials>
</Author>
```

by removing lower-case letters, spaces, and periods.

Nucleotide

Coding Sequences

What are the coding sequences in the *Escherichia coli* lac operon?

```
efetch -db nuccore -id J01636.1 -format gbc |
xtract -insd CDS gene sub_sequence
```

Sequence under a feature location is obtained with the −insd "sub_sequence" argument:

```
J01636.1    lacI    GTGAAACCAGTAACGTTATACGATGTCGCAGAGTATGCCG...
J01636.1    lacZ    ATGACCATGATTACGGATTCACTGGCCGTCGTTTTACAAC...
J01636.1    lacY    ATGTACTATTTAAAAAACACAAACTTTTGGATGTTCGGTT...
J01636.1    lacA    TTGAACATGCCAATGACCGAAAGAATAAGAGCAGGCAAGC...
```

Untranslated Region Sequences

What are the 5' and 3' UTR sequences for lycopene cyclase mRNAs?

```
SubSeq() {

  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 \
        -first INSDInterval_from -last INSDInterval_to -element "&PRD" "&SEQ"
}

UTRs() {

  while IFS=$'\t' read acc fst lst prd seq
  do
    if [ $fst -gt 1 ]
    then
      echo -e ">$acc 5'UTR: 1..$((fst-1)) $prd"
      echo "${seq:1:$((fst-2))}" | fold -w 50
    else
      echo -e ">$acc NO 5'UTR"
    fi
    if [ $lst -lt ${#seq} ]
    then
      echo -e ">$acc 3'UTR: $((lst+1))..${#seq} $prd"
      echo "${seq:$lst}" | fold -w 50
```

```
    else
      echo -e ">$acc NO 3'UTR"
    fi
  done
}

esearch -db nuccore -query "5.5.1.19 [ECNO]" |
efilter -molecule mrna -feature cds -source refseq |
efetch -format gbc |
SubSeq | UTRs
```

Sequences before the start codon and after the stop codon are obtained with Unix substring commands:

```
>NM_001324787.1 NO 5'UTR
>NM_001324787.1 NO 3'UTR
>NM_001324979.1 5'UTR: 1..262 lycopene beta cyclase, chloroplastic/chromoplastic
attatagaaatacttaagatatatcattgccctttaatcatttatttta
actcttttaagtgtttaaagattgattctttgtacatgttctgcttcatt
tgtgttgaaaattgagttgttttcttgaattttgcaagaatataggggac
cccatttgtgttgaaaattgagcagctttctttgtgttttgttcgatttt
tcaagaatataggaccccatttctgttttcttgagataaattgcacctt
gttgggaaaat
>NM_001324979.1 3'UTR: 1760..1782 lycopene beta cyclase, chloroplastic/chromoplastic
attcgacttatctgggatcttgt
...
```

5-Column Feature Table

How can you generate a 5-column feature table from GenBank format?

```
XtoT() {

  xtract -pattern INSDSeq -pfx ">Feature " \
      -first INSDSeqid,INSDSeq_accession-version \
    -group INSDFeature -FKEY INSDFeature_key \
      -block INSDInterval -deq "\n" \
        -element INSDInterval_from INSDInterval_to \
          INSDInterval_point INSDInterval_point \
          "&FKEY" -FKEY "()" \
      -block INSDQualifier -deq "\n\t\t\t" \
        -element INSDQualifier_name INSDQualifier_value
}

efetch -db nuccore -id U54469 -format gb |
transmute -g2x |
XtoT
```

Exploring the INSDSeq XML hierarchy with a −pattern {sequence} −group {feature} −block {qualifier} construct, and adding proper indentation, can produce feature table submission format:

```
>Feature U54469.1
1       2881    source
                        organism        Drosophila melanogaster
                        mol_type        genomic DNA
                        db_xref         taxon:7227
                        chromosome      3
                        map             67A8-B2
80      2881    gene
                        gene            eIF4E
```

```
80      224     mRNA
892     1458
1550    1920
1986    2085
2317    2404
2466    2881
                        gene            eIF4E
                        product         eukaryotic initiation factor 4E-I
                        ...
```

An **xml2tbl** script containing this xtract command is now included with EDirect.

WGS Components

How can you get FASTA format for all components of a WGS project?

```
GenerateWGSAccessions() {

  gb=$( efetch -db nuccore -id "$1" -format gbc < /dev/null )
  ln=$( echo "$gb" | xtract -pattern INSDSeq -element INSDSeq_length )
  pf=$(
    echo "$gb" |
    xtract -pattern INSDSeq -element INSDSeq_locus |
    sed -e 's/010*/01/g'
  )
  seq -f "${pf}%07.0f" 1 "$ln"
}

GenerateWGSAccessions "JABRPF000000000" |
```

returns the expanded list of accessions from the WGS master:

```
JABRPF010000001
JABRPF010000002
JABRPF010000003
...
JABRPF010000061
JABRPF010000062
JABRPF010000063
```

Piping those accessions to:

```
efetch -db nuccore -format fasta
```

retrieves the individual components in FASTA format:

```
>JABRPF010000001.1 Enterococcus faecalis strain G109-2 contig00001, ...
ACACTAATATGTGTCTTTTTAGACACTAGCTCACTAAAAAAAATAGTCATAATTTCTTCATTATTAAAAT
CCAACAATTGTGAAATCAATTTAATATCCGATGCTTTGAAAACAACTTCTCCTTTTAATTTTTTGTAAAT
CGTTGAAGCGGATATTGGTTGCCCGTATGCAGTCATTTCATTTGCCAACCACTCAACATTTTTTCCTTTC
...
```

Proteins in a Region

How do you get the protein records encoded in a specific region of a nucleotide sequence?

```
efetch -db nuccore -id NC_015162.1 -format gb -seq_start 9125 -seq_stop 103803 |
xtract -insd CDS protein_id |
cut -f 2 |
efetch -db protein -format fasta
```

extracts the protein_id accessions and retrieves those proteins in FASTA format:

```
>WP_013615770.1 hypothetical protein [Deinococcus proteolyticus]
MTQNASAGLTAETLMTETGLSAAAVRRALKAYDAAFGLEHPTQDGELHLTPAEYDVLRRALQLTGGYAPG
LKLWFGEQQALALSTQPVASPREVQQLSQLYQQVESYRARPPEEPAQTLRALLDQAQALGWSGFWEMGRL
QGAALFVLGVLRFEDGQRAKVSMQTPLSGAEALVLSRGVCALLQRVRTQPGSGQAWSWNEVLLEEVERIL
KDLTE
>WP_013615771.1 hypothetical protein [Deinococcus proteolyticus]
MNKDTNVDTAFAGGFFTTFENGECVFYDADGVCHSVTPDTVAAVVASGTPRDVVALHEDVQGRTDAVAQV
...
```

Reverse Complement GenBank

How do you reverse complement a nucleotide sequence in GenBank format?

```
efetch -db nuccore -id J01749 -format gb |
gbf2fsa |
transmute -revcomp |
transmute -fasta -width 50
```

converts GenBank to FASTA, reverse-complements the sequence, and saves it at the selected 50 characters per line:

```
TTCTTGAAGACGAAAGGGCCTCGTGATACGCCTATTTTTATAGGTTAATG
TCATGATAATAATGGTTTCTTAGACGTCAGGTGGCACTTTTCGGGGAAAT
GTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATTCAAATATGTA
...
ACGATGAGCGCATTGTTAGATTTCATACACGGTGCCTGACTGCGTTAGCA
ATTTAACTGTGATAAACTACCGCATTAAAGCTTATCGATGATAAGCTGTC
AAACATGAGAA
```

Reverse Complement NCBI2NA

How do you reverse complement a 2-bit encoded nucleotide sequence?

```
efetch -db nuccore -id J01749 -format asn |
xtract -pattern Seq-entry \
  -group seq -if ncbi2na \
    -block inst -LEN length -SEQ -ncbi2na ncbi2na \
      -SUB "&SEQ[:&LEN]" -REV -revcomp "&SUB" \
      -sep "\n" -fasta "&REV"
```

expands the NCBI2NA representation to IUPACNA, truncates at the indicated length, reverse-complements that sequence, and saves it at the default 70 characters per line:

```
TTCTTGAAGACGAAAGGGCCTCGTGATACGCCTATTTTTATAGGTTAATGTCATGATAATAATGGTTTCT
TAGACGTCAGGTGGCACTTTTCGGGGAAATGTGCGCGGAACCCCTATTTGTTTATTTTTCTAAATACATT
CAAATATGTATCCGCTCATGAGACAATAACCCTGATAAATGCTTCAATAATATTGAAAAAGGAAGAGTAT
...
GCATAACCAAGCCTATGCCTACAGCATCCAGGGTGACGGTGCCGAGGATGACGATGAGCGCATTGTTAGA
TTTCATACACGGTGCCTGACTGCGTTAGCAATTTAACTGTGATAAACTACCGCATTAAAGCTTATCGATG
ATAAGCTGTCAAACATGAGAA
```

Six-Frame Protein Translation

How do you translate all six reading frames of a nucleotide sequence?

```
efetch -db nuccore -id J01749 -format fasta |
transmute -cds2prot -circular -gcode 11 -all
```

will simultaneously translate three plus-strand and three minus-strand frames, including codons that span the origin if the molecule is circular:

```
>J01749.1-1+
FSCLTAYHR*ALMR*FITVKLLTQSGTVYE ... KCHLTSKKPLLS*H*PIKIGVSRGPFVFKN
>J01749.1-2+
SHV*QLIIDKL*CGSLSQLNC*RSQAPCMK ... SAT*RLRNHYYHDINL*K*AYHEALSSSRI
>J01749.1-3+
LMFDSLSSISFNAVVYHS*IANAVRHRV*N ... VPPDV*ETIIIMTLTYKNRRITRPFRLQEF
>J01749.1-1-
ILEDERAS*YAYFYRLMS***WFLRRQVAL ... FIHGA*LR*QFNCDKLPH*SLSMISCQT*E
>J01749.1-2-
NS*RRKGLVIRLFL*VNVMIIMVS*TSGGT ... FHTRCLTALAI*L**TTALKLIDDKLSNMR
>J01749.1-3-
EFLKTKGPRDTPIFIG*CHDNNGFLDVRWH ... ISYTVPDCVSNLTVINYRIKAYR**AVKHE
```

Pattern Searching

The pBR322 cloning vector is a circular plasmid with unique restriction sites in two antibiotic resistance genes.

The transmute -replace function introduces a second BamHI restriction enzyme recognition site by modifying the ribosomal binding site of the pBR322 rop (restrictor of plasmid copy number) gene:

```
efetch -db nuccore -id J01749 -format fasta |
transmute -replace -offset 1907 -delete GG -insert TC |
```

The transmute -search function takes a list of sequence patterns with optional labels (such as restriction enzyme names), and uses a finite-state algorithm to simultaneously search for all patterns:

```
transmute -search -circular GGATCC:BamHI GAATTC:EcoRI CTGCAG:PstI |
align-columns -g 4 -a rl
```

The (0-based) starting positions and labels for each match are then printed in a two-column table:

```
 374    BamHI
1904    BamHI
3606    PstI
4358    EcoRI
```

The **disambiguate-nucleotides** and **systematic-mutations** scripts can generate all possible single-base substitutions in a pattern for a more relaxed search.

Protein

Amino Acid Composition

What is the amino acid composition of human titin?

```
#!/bin/bash -norc

AAComp() {

  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 )

  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
}

efetch -db protein -id Q8WZ42 -format gpc |
xtract -pattern INSDSeq -element INSDSeq_sequence |
AAComp
```

This produces a table of residue counts using the three-letter amino acid abbreviations:

```
Ala    2084
Arg    1640
Asn    1111
Asp    1720
Cys    513
Gln    942
Glu    3193
Gly    2066
His    478
Ile    2062
Leu    2117
Lys    2943
Met    398
Phe    908
Pro    2517
Ser    2463
Thr    2546
Trp    466
Tyr    999
Val    3184
```

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, sorts by sequence length, and saves the GenPept results to individual files named by their sequence accessions, using the right angle bracket (">") Unix output redirection character:

```
EFN61235.gpf
EFN80340.gpf
EGW08477.gpf
EKC18433.gpf
ELK28555.gpf
```

Archaea Enzyme

Which archaebacteria have chloramphenicol acetyltransferase?

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

Filtering on the definition line produces a list of archaeal organism names:

```
Euryarchaeota archaeon
Methanobrevibacter arboriphilus
Methanobrevibacter gottschalkii
Methanobrevibacter millerae
Methanobrevibacter oralis
...
```

Gene

Gene Counts

How many genes are on each human chromosome?

```
for chr in {1..22} X Y MT
do
  esearch -db gene -query "Homo sapiens [ORGN] AND $chr [CHR]" |
  efilter -status alive -type coding |
  efetch -format docsum |
  xtract -pattern DocumentSummary -NAME Name \
    -block GenomicInfoType -if ChrLoc -equals "$chr" \
      -tab "\n" -element ChrLoc,"&NAME" |
  sort | uniq | cut -f 1 |
  sort-uniq-count-rank |
  reorder-columns 2 1
done
```

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

```
1     2011
2     1224
3     1046
4     742
5     854
6     1015
7     911
8     666
9     763
10    718
11    1279
12    1012
13    327
14    601
15    585
16    835
17    1147
18    266
```

```
19    1391
20    528
21    232
22    429
X     839
Y     63
MT    13
```

The range construct cannot be used for Roman numerals, so the equivalent query on *Saccharomyces cerevisiae* would need to explicitly list all chromosomes, including the mitochondrion:

```
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]" or −location plastid.

Chromosome Locations

Where are mammalian calmodulin genes located?

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

The −def command adds a dash to prevent missing data from shifting columns in the table:

```
801       CALM1     14q32.11     Homo sapiens
808       CALM3     19q13.32     Homo sapiens
805       CALM2     2p21         Homo sapiens
24242     Calm1     6q32         Rattus norvegicus
12313     Calm1     12 E         Mus musculus
50663     Calm2     6q12         Rattus norvegicus
24244     Calm3     1q21         Rattus norvegicus
12315     Calm3     7 9.15 cM    Mus musculus
12314     Calm2     17 E4        Mus musculus
80796     Calm4     13 A1        Mus musculus
617095    CALM1     -            Bos taurus
396838    CALM3     -            Sus scrofa
520277    CALM3     -            Bos taurus
364774    Calml5    17q12.2      Rattus norvegicus
...
```

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

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

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

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

Final filtering and sorting:

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

results in a table of exon counts and transcript lengths:

```
NC_000023.11    79    NM_004010.3       14083    Dp427p2
NC_000023.11    79    NM_004009.3       14000    Dp427p1
NC_000023.11    79    NM_004006.3       13992    Dp427m
NC_000023.11    79    NM_000109.4       13854    Dp427c
NC_000023.11    78    XM_006724468.2    13923    X1
NC_000023.11    78    XM_017029328.1    13916    X4
NC_000023.11    78    XM_006724469.3    13897    X2
NC_000023.11    77    XM_006724470.3    13875    X3
...
```

Upstream Sequences

What sequences are upstream of phenylalanine hydroxylase genes?

```
esearch -db nuccore -query "U49897 [ACCN]" |
elink -target gene |
elink -target homologene |
elink -target gene |
efetch -format docsum |
xtract -pattern DocumentSummary -if GenomicInfoType -element Id \
  -block GenomicInfoType -element ChrAccVer -1-based ChrStart ChrStop |
```

This produces a table with 1-based sequence coordinates:

```
5053       NC_000012.12    102958441    102836889
18478      NC_000076.7     87357657     87419999
38871      NT_037436.4     7760453      7763166
24616      NC_005106.4     28066639     28129772
378962     NC_007115.7     17409367     17391680
...
```

Then, 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:

```
upstream.sh 500
```

to extract and print the 500 nucleotides immediately upstream of each gene:

```
>NC_000012.12:c102958941-102958442 Homo sapiens chromosome 12, GRCh38.p13 ...
TGAAGTCGAGAAGCTCCTGCTCCTCGGGGCTGAGCGGGTCGTAAGAGCCCTCGTCCGACGAGTAGGATGA
GACCGGCGAGCCGGCCATGGAGTTCAAGTCGTTGGAGTAGTTGGGGGAGATGGTGGGCGACAGGACGCCT
GCCTGGAAGGCGGCGCTCACCGCGTCATGCTCGTCCAGCAGCTGCTGCAGCGCGCGGATGTACTCGACCG
...
```

Assembly

Complete Genomes

What complete genomes are available for *Escherichia coli*?

```
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-table -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_002695.2    5498578    Escherichia coli O157:H7 str. Sakai DNA
NC_000913.3    4641652    Escherichia coli str. K-12 substr. MG1655
NC_002128.1    92721      Escherichia coli O157:H7 str. Sakai plasmid pO157
NC_002127.1    3306       Escherichia coli O157:H7 str. Sakai plasmid pOSAK1
```

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

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

```
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 | grep -v patch |
sort
```

returns the assembled chromosome and mitochondrial sequence records:

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

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

```
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-table -k 2,2nr
done
```

which generates:

```
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    5227293    Bacillus anthracis str. Ames chromosome
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_002946.2    2153922    Neisseria gonorrhoeae FA 1090 chromosome
NC_002516.2    6264404    Pseudomonas aeruginosa PAO1 chromosome
NC_004337.2    4607202    Shigella flexneri 2a str. 301 chromosome
NC_003028.3    2160842    Streptococcus pneumoniae TIGR4 chromosome
NC_003098.1    2038615    Streptococcus pneumoniae R6 chromosome
```

SNP

SNP Data Table

How can you obtain a tab-delimited table of SNP attributes from a document summary?

```
efetch -db snp -id 11549407 -format docsum |
snp2tbl
```

The **snp2tbl** script extracts fields from HGVS data and prints the individual values for further processing:

```
rs11549407    NC_000011.10    5226773    G    A    Genomic    Substitution    HBB
rs11549407    NC_000011.10    5226773    G    C    Genomic    Substitution    HBB
rs11549407    NC_000011.10    5226773    G    T    Genomic    Substitution    HBB
rs11549407    NG_000007.3     70841      C    A    Genomic    Substitution    HBB
rs11549407    NG_000007.3     70841      C    G    Genomic    Substitution    HBB
rs11549407    NG_000007.3     70841      C    T    Genomic    Substitution    HBB
...
rs11549407    NM_000518.5     167        C    A    Coding     Substitution    HBB
rs11549407    NM_000518.5     167        C    G    Coding     Substitution    HBB
rs11549407    NM_000518.5     167        C    T    Coding     Substitution    HBB
rs11549407    NP_000509.1     39         Q    *    Protein    Termination     HBB
rs11549407    NP_000509.1     39         Q    E    Protein    Missense        HBB
rs11549407    NP_000509.1     39         Q    K    Protein    Missense        HBB
```

Columns are SNP identifier, accession.version, offset from start of sequence, letters to delete, letters to insert, sequence class, variant type, and gene name. The internal steps (**snp2hgvs**, **hgvs2spdi**, and **spdi2tbl**) are described below.

Amino Acid Substitutions

What are the missense products of green-sensitive opsin?

```
esearch -db gene -query "OPN1MW [PREF] AND human [ORGN]" |
elink -target snp | efilter -class missense |
efetch -format docsum |
```

SNP document summaries contain HGVS data of the form:

```
NC_000023.11:g.154193517C>A, ... ,NP_000504.1:p.Ala285Val
```

This can be parsed by an xtract **–hgvs** command within the **snp2hgvs** script:

```
snp2hgvs |
```

into a structured representation of nucleotide and amino acid replacements:

```
...
<HGVS>
  <Id>782327292</Id>
  <Gene>OPN1MW</Gene>
  <Variant>
    <Class>Genomic</Class>
    <Type>Substitution</Type>
```

```
      <Accession>NC_000023.11</Accession>
      <Position>154193516</Position>
      <Deleted>C</Deleted>
      <Inserted>A</Inserted>
      <Hgvs>NC_000023.11:g.154193517C&gt;A</Hgvs>
    </Variant>
    ...
    <Variant>
      <Class>Coding</Class>
      <Type>Substitution</Type>
      <Accession>NM_000513.2</Accession>
      <Offset>853</Offset>
      <Deleted>C</Deleted>
      <Inserted>T</Inserted>
      <Hgvs>NM_000513.2:c.854C&gt;T</Hgvs>
    </Variant>
    <Variant>
      <Class>Protein</Class>
      <Type>Missense</Type>
      <Accession>NP_000504.1</Accession>
      <Position>284</Position>
      <Deleted>A</Deleted>
      <Inserted>D</Inserted>
      <Hgvs>NP_000504.1:p.Ala285Asp</Hgvs>
    </Variant>
    <Variant>
      <Class>Protein</Class>
      <Type>Missense</Type>
      <Accession>NP_000504.1</Accession>
      <Position>284</Position>
      <Deleted>A</Deleted>
      <Inserted>V</Inserted>
      <Hgvs>NP_000504.1:p.Ala285Val</Hgvs>
    </Variant>
  </HGVS>
  ...
```

where the original 1-based HGVS positions are converted to 0-based in the XML.

Passing those results through the **hgvs2spdi** script:

```
  hgvs2spdi |
```

converts CDS-relative offsets to sequence-relative positions:

```
  ...
  <SPDI>
    <Id>782327292</Id>
    <Gene>OPN1MW</Gene>
    ...
    <Variant>
      <Class>Coding</Class>
      <Type>Substitution</Type>
      <Accession>NM_000513.2</Accession>
      <Position>935</Position>
      <Offset>853</Offset>
      <Deleted>C</Deleted>
      <Inserted>A</Inserted>
      <Hgvs>NM_000513.2:c.854C&gt;A</Hgvs>
      <Spdi>NM_000513.2:935:C:A</Spdi>
```

```
    </Variant>
    <Variant>
      <Class>Coding</Class>
      <Type>Substitution</Type>
      <Accession>NM_000513.2</Accession>
      <Position>935</Position>
      <Offset>853</Offset>
      <Deleted>C</Deleted>
      <Inserted>T</Inserted>
      <Hgvs>NM_000513.2:c.854C&gt;T</Hgvs>
      <Spdi>NM_000513.2:935:C:T</Spdi>
    </Variant>
    ...
  </SPDI>
  ...
```

Piping this through the **spdi2tbl** script:

```
spdi2tbl |
```

completes the final step of the **snp2tbl** process to generate a SNP Data Table. Filtering that through:

```
grep Protein | grep Missense | cut -f 1-5
```

results in a table of amino acid substitutions sorted by accession, position, and residue:

```
rs1238141906    NP_000504.1    40    E    K
rs1189783086    NP_000504.1    42    P    L
rs1257135801    NP_000504.1    45    H    Y
rs1284438666    NP_000504.1    63    V    I
rs1223726997    NP_000504.1    64    I    T
...
```

Those rows are processed in a while loop that caches the current sequence data:

```
while read rsid accn ofs del ins
do
  if [ "$accn" != "$last" ]
  then
    seq=$( efetch -db protein -id "$accn" -format gpc < /dev/null |
           xtract -pattern INSDSeq -lower INSDSeq_sequence )
    last="$accn"
  fi
  pos=$((ofs + 1))
  echo ">$rsid [$accn $ins@$pos]"
  echo "$seq" |
  transmute -replace -offset "$ofs" -delete "$del" -insert "$ins" -lower |
  fold -w 50
done
```

and uses transmute **–replace** to generate modified FASTA with substituted residues in upper case:

```
>rs1238141906 [NP_000504.1 K@41]
maqqwslqrlagrhpqdsyedstqssiftytnsnstrgpfKgpnyhiapr
wvyhltsvwmifvviasvftnglvlaatmkfkklrhplnwilvnlavadl
aetviastisvvnqvygyfvlghpmcvlegytvslcgitglwslaiiswe
...
```

SNP-Modified Product Pairs

How can you match codon modifications with amino acid substitutions?

```
efetch -db snp -id 11549407 -format docsum |
snp2tbl |
tbl2prod
```

For SNPs that have different substitutions at the same position, the **tbl2prod** script translates coding sequences (after nucleotide modification), and sorts them with protein sequences (after residue replacement), to produce adjacent matching CDS/protein pairs:

```
rs11549407    NM_000518.5:167:C:T    MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWT*R...
rs11549407    NP_000509.1:39:Q:*     MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWT*R...
rs11549407    NM_000518.5:167:C:G    MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTER...
rs11549407    NP_000509.1:39:Q:E     MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTER...
rs11549407    NM_000518.5:167:C:A    MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTKR...
rs11549407    NP_000509.1:39:Q:K     MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTKR...
rs11549407    NM_000518.5:167:C:+    MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQR...
rs11549407    NP_000509.1:39:Q:+     MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQR...
```

The "+" sign indicates the unmodified "wild-type" nucleotide or amino acid.

Structure

Structural Similarity

What archaea structures are similar to snake venom phospholipase?

```
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
```

Structure neighbors use geometric comparison 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

Taxonomic Lineage

What are the major taxonomic lineage nodes for humans?

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

This uses the double star / child construct to recursively explore the data hierarchy:

```
9606            Homo sapiens
superkingdom    Eukaryota
kingdom         Metazoa
```

```
phylum          Chordata
subphylum       Craniata
superclass      Sarcopterygii
class           Mammalia
superorder      Euarchontoglires
order           Primates
...
```

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
...
Zingiber spectabile
Zygnema circumcarinatum
```

BioSample

BioSample document summaries:

```
esummary -db biosample -id SAMN34375013 |
```

use XML attributes to identify data element types:

```
<Attribute attribute_name="strain" harmonized_name="strain" ...>KF24</Attribute>
```

A two-stage xtract pipeline can produce a tab-delimited table, with one column for each selected field. Piping the data to the first command:

```
xtract -rec BioSampleInfo -pattern DocumentSummary \
  -wrp Accession -element Accession \
  -wrp Title -element DocumentSummary/Title \
  -wrp Link -sep "|" -numeric Links/Link \
  -group Attribute -if @harmonized_name \
    -TAG -lower @harmonized_name -wrp "&TAG" -element Attribute |
```

generates an intermediate form, with XML object names taken from the "harmonized_name" attributes:

```
<BioSampleInfo>
  <Accession>SAMN34375013</Accession>
  <Title>Microbe sample from Bacillus subtilis</Title>
  <Link>960711</Link>
  <isolation_source>rhizosphere soil</isolation_source>
  <collection_date>missing</collection_date>
  <geo_loc_name>China: Kaifeng, Henan Province</geo_loc_name>
  <sample_type>bacterial isolate</sample_type>
```

```
    <lat_lon>34.14 N 114.05 E</lat_lon>
    <strain>KF24</strain>
  </BioSampleInfo>
```

(A **bsmp2info** script containing this first xtract command is now included with EDirect.)

Desired fields can then be selected by name in the last line of the second xtract command, with a "-" placeholder printed for any missing values:

```
xtract -pattern BioSampleInfo -def "-" -first Accession Title Link \
  geo_loc_name isolation_source strain lat_lon
```

Using −first instead of −element eliminates possible redundant entries during the attribute name transition from "country" to "geo_loc_name".

SRA

Using RunInfo Format

SRA data can be retrieved in RunInfo format:

```
efetch -db sra -id SRR6314034 -format runinfo |
```

as comma-separated values, with the first line containing the field names:

```
Run,ReleaseDate,LoadDate,spots,bases,spots_with_mates,avgLength,...
SRR6314034,2017-11-21 23:27:11,2017-11-21 23:25:38,128,539118,0,...
```

Piping to the **csv2xml** script, and using the −header flag:

```
csv2xml -set Set -rec Rec -header |
```

converts the data into XML:

```
<Set>
  <Rec>
    <Run>SRR6314034</Run>
    <ReleaseDate>2017-11-21 23:27:11</ReleaseDate>
    <LoadDate>2017-11-21 23:25:38</LoadDate>
    <spots>128</spots>
    <bases>539118</bases>
    <spots_with_mates>0</spots_with_mates>
    <avgLength>4211</avgLength>
    <size_MB>0</size_MB>
    <AssemblyName></AssemblyName>
    <download_path>...</download_path>
    <Experiment>SRX3413965</Experiment>
    <LibraryName>child</LibraryName>
    <LibraryStrategy>AMPLICON</LibraryStrategy>
    <LibrarySelection>PCR</LibrarySelection>
    <LibrarySource>GENOMIC</LibrarySource>
    <LibraryLayout>SINGLE</LibraryLayout>
    <InsertSize>0</InsertSize>
    <InsertDev>0</InsertDev>
    <Platform>PACBIO_SMRT</Platform>
    <Model>PacBio RS II</Model>
    <SRAStudy>SRP125431</SRAStudy>
    <BioProject>PRJNA418990</BioProject>
    <Study_Pubmed_id></Study_Pubmed_id>
    <ProjectID>418990</ProjectID>
```

```
      <Sample>SRS2707133</Sample>
      <BioSample>SAMN08040264</BioSample>
      <SampleType>simple</SampleType>
      <TaxID>9606</TaxID>
      <ScientificName>Homo sapiens</ScientificName>
      <SampleName>BBS9_del_mother</SampleName>
      <g1k_pop_code></g1k_pop_code>
      <source></source>
      <g1k_analysis_group></g1k_analysis_group>
      <Subject_ID></Subject_ID>
      <Sex>female</Sex>
      <Disease></Disease>
      <Tumor>no</Tumor>
      <Affection_Status></Affection_Status>
      <Analyte_Type></Analyte_Type>
      <Histological_Type></Histological_Type>
      <Body_Site></Body_Site>
      <CenterName>ICAHN SCHOOL OF MEDICINE AT MOUNT SINAI</CenterName>
      <Submission>SRA633054</Submission>
      <dbgap_study_accession></dbgap_study_accession>
      <Consent>public</Consent>
      <RunHash>19AC4EB8A65D733274756464DCCF65EA</RunHash>
      <ReadHash>AC8F39C51F95D9CD1BD0CBFBB669AD1E</ReadHash>
    </Rec>
  </Set>
```

This can then be piped through xtract:

```
xtract -pattern Rec -def "-" -element Run Experiment BioProject BioSample
```

to retrieve the desired values by field name:

```
SRR6314034    SRX3413965    PRJNA418990    SAMN08040264
```

Installation of EDirect on Cloud

To install the EDirect software, open a terminal window and execute the following command:

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

At the end of installation, answer "y" to have the script run the PATH update command to edit your configuration file, so that EDirect programs can be run in subsequent terminal sessions.

One installation is complete, run:

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

to set the PATH for the current terminal session.

Downloading BLAST Software

Obtain the Magic-BLAST software if it is not already installed:

```
download-ncbi-software magic-blast
```

Preparing Chromosome Files

Retrieve the sequence of human chromosome 7 with:

```
efetch -db nuccore -id NC_000007 -format fasta -immediate > NC_000007.fsa
```

Download protein-coding genes on human chromosome 7 in document summary format, and extract the gene ranges into a tab-delimited table:

```
esearch -db gene -query "Homo sapiens [ORGN] AND 7 [CHR]" |
efilter -status alive -type coding | efetch -format docsum |
gene2range "7" > NC_000007.gen
```

SRA Gene Analysis

Run a **magicblast** search of SRR6314034 against chromosome 7:

```
magicblast -sra SRR6314034 -subject NC_000007.fsa -outfmt asn |
asn2xml > SRR6314034.xml
```

Passing the alignment details through the **blst2tkns** script:

```
cat SRR6314034.xml |
blst2tkns |
```

tokenizes the alignment parts:

```
index          1
score          6268
start          33088037
stop           33167397
strand         plus
match          12
mismatch       1
genomic-ins    1
...
```

Piping these to the **split-at-intron** script:

```
split-at-intron |
```

detects large genomic insertions:

```
6268    1    plus    33088037..33091003,33163798..33167397
4910    2    plus    33089462..33091003,33163799..33167397
1814    3    plus    33164965..33167397
...
```

These are filtered by minimum score with:

```
filter-columns '$1 > 1000' |
```

The **fuse-ranges** script then merges overlapping alignments:

```
fuse-ranges |
```

into a minimal set of extended ranges:

```
plus     33088037    33091003    2967
plus     33163798    33167398    3601
plus     33272248    33272278    31
plus     33394238    33394551    314
minus    33088037    33091004    2968
minus    33163798    33167398    3601
minus    33255886    33256012    127
minus    33394326    33394551    226
```

Running each range through the **find-in-gene** script:

```
while read std min max len
do
  cat NC_000007.gen |
  find-in-gene "$std" "$min" "$max"
done |
sort -f | uniq -i
```

returns the names of gene(s) that overlap the aligned segments:

```
BBS9
```

PubChem

PubChem in Entrez

Entrez can search on the complete synonym of a compound:

```
esearch -db pccompound -query "catechol [CSYN]" |
efetch -format uid
```

to return a small number of closely matching compound identifiers (CIDs):

```
73160
9064
289
```

Entrez document summaries for a compound:

```
efetch -db pccompound -id 289 -format docsum |
xtract -pattern DocumentSummary -element MolecularFormula IsomericSmiles
```

contains general descriptive information fields for the compound:

```
C6H6O2    C1=CC=C(C(=C1)O)O
```

Power User Gateway (PUG) REST Query Form

PubChem also supports a RESTful service for more advanced queries. Nquire provides a −pugrest URL shortcut. The base form of a search request is is:

```
nquire -pugrest [compound|substance|assay] [input] [operation] [output]
```

Searches can use the name of a compound:

```
nquire -pugrest compound name catechol cids TXT
```

to obtain the best matching compound identifier:

```
289
```

The CID can be used as the input key to obtain a title and description:

```
nquire -pugrest compound cid 289 description XML
```

or to retrieve a much more detailed record:

```
nquire -pugrest compound cid 289 record XML |
```

that includes the canonical or isomeric SMILES codes:

```
xtract -pattern PC-InfoData \
  -if PC-Urn_label -equals SMILES -and PC-Urn_name -equals Isomeric \
    -element PC-InfoData_value_sval
```

PubChem Chemical Identifiers

Certain identifier types require POST arguments to encode special symbols:

```
nquire -pugrest compound smiles description XML \
  -smiles "C1=CC=C(C(=C1)O)O" |
xtract -pattern InformationList -element Title Description
```

This returns the chemical name and description:

```
Catechol    Catechol is a benzenediol comprising...
```

Other identifier key types that are encoded in separate arguments are shown below:

```
nquire -pugrest compound inchi synonyms TXT \
  -inchi "1S/C6H6O2/c7-5-3-1-2-4-6(5)8/h1-4,7-8H"

nquire -pugrest compound inchikey cids JSON \
  -inchikey "YCIMNLLNPGFGHC-UHFFFAOYSA-N"

nquire -pugrest compound/fastsubstructure/smarts/cids/XML \
  -smarts "[#7]-[#6]-1=[#6]-[#6](C#C)=[#6](-[#6]-[#8])-[#6]=[#6]-1"
```

(Nquire −inchi will supply the expected "InChI=" prefix if it is missing in the argument string.)

PUG-REST Asynchronous Queries

Some PUG-REST queries are computationally intensive and run asynchronously:

```
nquire -pugrest compound/superstructure/cid/2244/XML |
```

The returned <ListKey> token is piped to an nquire −pugwait command, which polls the server until the results are available:

```
nquire -pugwait
```

Identifiers are then downloaded and placed directly in an ENTREZ_DIRECT message:

```
<ENTREZ_DIRECT>
  <Db>pccompound</Db>
  <Count>3750</Count>
  <Id>87</Id>
  <Id>175</Id>
  <Id>176</Id>
  ...
  <Id>162400221</Id>
  <Id>162416056</Id>
  <Id>162417911</Id>
</ENTREZ_DIRECT>
```

Support for this form was added when EDirect was redesigned in 2020.

Bibliometrics

Reverse Chronological Order

Repackaging an entire document summary and creating a set of keys for sorting:

```
esearch -db pubmed -query "tn3 transposition immunity" |
efetch -format docsum |
xtract -rec Rec -pattern DocumentSummary -INDX "+" \
  -group DocumentSummary -pkg SortKeys \
    -unit DocumentSummary -wrp INDX -element "&INDX" \
    -unit PubDate -wrp YEAR -year PubDate \
    -unit PubDate -wrp DATE -date "*" \
    -unit Title -wrp TITL -lower Title \
    -unit Author -position first -wrp FAUT -lower Name \
    -unit Author -position last -wrp LAUT -lower Name \
    -unit Authors -wrp ANUM -num Author/Name \
    -unit DocumentSummary -wrp SIZE -len "*" \
  -group DocumentSummary -pkg DS -element "*" |
```

produces an intermediate structure with separate containers for the sort keys and the original docsum:

```
...
<Rec>
  <SortKeys>
    <INDX>5</INDX>
    <YEAR>1989</YEAR>
    <DATE>1989/04</DATE>
    <TITL>nucleotide sequences required for tn3 transposition immunity.</TITL>
    <FAUT>kans ja</FAUT>
    <LAUT>casadaban mj</LAUT>
    <ANUM>2</ANUM>
    <SIZE>1695</SIZE>
  </SortKeys>
  <DS>
    <DocumentSummary>
      <Id>2539356</Id>
      <PubDate>1989 Apr</PubDate>
      <Source>J Bacteriol</Source>
      ...
    </DocumentSummary>
  </DS>
</Rec>
...
```

This allows sorting of records by a new tag derived from −year:

```
xtract -rec Rec -pattern Rec -sort-rev SortKeys/YEAR |
xtract -set DocumentSummarySet -pattern Rec \
  -group DS/DocumentSummary -element "*" |
transmute -format -combine |
xtract -pattern DocumentSummary -element Id PubDate
```

to present publication records with the most recent shown first:

```
36257990    2022 Oct 18
28096365    2017 Jan 31
22624153    2012 Feb
```

```
21729108    2011 Sep
8595595     1996 Jan
```

Count Unique Journals

Some bibliometric analyses require fetching huge numbers of PubMed records. One such example is counting the number of unique journals publishing papers indexed under a given category.

A common function consolidates code to execute a query and fetch the records from the EDirect local PubMed archive. The xtract **−histogram** shortcut acts as an −element command followed by a built-in sort-uniq-count. (This can save significant time when the number of results is in the millions.) The line count gives the number of journals publishing at least one paper that satisfies the query:

```
CountUniqueJournals() {

  qry="$1"

  # count number of unique journals publishing papers that match a query
  phrase-search -db pubmed -query "$qry" | fetch-pubmed |
  xtract -pattern PubmedArticle -histogram Journal/ISOAbbreviation |
  wc -l | tr -d ' '
}
```

Looping through a series of years, and calling this function with and without a filter of interest in the query:

```
for year in {2016..2022}
do

  # want to find percent of journals still using unstructured date
  filt="medline date [PROP]"
  base="journal article [PTYP] AND $year [YEAR]"

  # subset of journals publishing a paper with filter condition
  subs=$( CountUniqueJournals "$base AND $filt" )

  # number of unique journals in selected publication types
  totl=$( CountUniqueJournals "$base" )

  frac="-"
  if [ "$totl" -gt 0 ]
  then
    # calculate (integer) percentage
    frac=$((subs * 100 / totl))
  fi

  # print year, journal counts, and percent using filter condition
  printf "$year\t$subs\t$totl\t$frac\n"

done | align-columns -h 2 -g 4 -a r
```

returns the year, the numbers of unique journals with the filter and in total for that year, and the ratio of the two as a percentage. In this example the last column shows a steady decrease in the percentage of journals providing an unstructured publication date:

```
2016    1933    10362    18
2017    1153    10473    11
2018    1071    10321    10
2019     980    10027     9
2020     919    10579     8
```

```
2021     950    10947     8
2022     835    10600     7
```