Bioinformatics Web Resources
NCBI / EBI / Uniprot / Pfam

Biol4230    Thurs, Feb 16, 2017
Bill Pearson  wrp@virginia.edu    4-2818    Jordan 6-057

- Recognizing web addresses (URLs)
- NCBI – eutilities: esearch/efetch/blast search
  www.ncbi.nlm.nih.gov/books/NBK25501/
- EBI – web services
  www.ebi.ac.uk/Tools/webservices/
- Uniprot ID mapper
  www.uniprot.org/faq/28#id_mapping_examples
- Pfam – using XML data
  pfam.xfam.org/help#tabview=tab10
  xml.etree.ElementTree

To learn more:

- Each of the web resources outlined has a help/FAQ page on downloading content
- Homework, due Monday, Feb. 20
See:  fasta.bioch.virginia.edu/biol4230/labs/accessions_hwk5.html
Questions include:
1. Do a text search at the NCBI and download all the human refseq protein accessions for "GSTM""
   • also store protein lengths (see NCBI XML slides)
2. Map each of the Refseq accessions to Uniprot accessions at the Uniprot ID mapping site
   • are all the human proteins present in Uniprot?
   • are the mapped proteins the same length?
   • are the mapped proteins the same identical sequence?
3. Look up the domain content for each of the Uniprot accessions in Pfam
   • For each of the human proteins that can be mapped to Uniprot and Pfam, how many of the proteins have Pfam domains that are less than 50% of the Pfam family model length?
URLs – HTTP requests
(how does the Web know what you want?)

• HTTP GET/PUT:
  http://fasta.bioch.virginia.edu/ web site
  fasta Www2/fasta Www.cgi?rm=select&pgm=fap
  script location/name.cgi
  beginning of arguments: ?arg1=value1
  - separator
  arg2=value2
  esearch.fcgi
  ?db=protein&term=GSTM*&rettype=uidlist&retmax=1000
  ?arg1=val1&arg2=val2&arg3=val3 (no spaces)

Information from the NCBI – eutils


• ESearch: responds to a text query with the list of UIDs matching the query in a
given database, along with the term translations of the query.
• ESummary: responds to a list of UIDs with the corresponding document
  summaries.
• EFetch: responds to a list of UIDs with the corresponding data records.
  [reference documentation]
• ELink: responds to a list of UIDs in a given database with either a list of related
  IDs in the same database or a list of linked IDs in another Entrez database.
• EGQuery: responds to a text query with the number of records matching the
  query in each Entrez database.

Getting NCBI data using eutils is ALWAYS a two step process:
1. get a set of accessions with Esearch
2. retrieve the data using the list of accessions
How to find data: NCBI

www.ncbi.nlm.nih.gov/books/NBK25500/
www.ncbi.nlm.nih.gov/books/NBK25497/

curl


<?xml version="1.0" encoding="UTF-8"?>
<eSearchResult>
  <Count>41</Count>
  <RetMax>41</RetMax>
  <RetStart>0</RetStart>
  <IdList>
    <Id>NP_001278092.1</Id>
    <Id>NP_001129490.1</Id>
    <Id>NP_758859.1</Id>
    <Id>NP_071900.2</Id>
    <Id>NP_000843.1</Id>
    <Id>NP_000111.1</Id>
    ... stuff deleted
    <Id>NP_000842.2</Id>
    <Id>NP_000841.1</Id>
    <Id>NP_671489.1</Id>
    <Id>NP_746543.1</Id>
    <Id>NP_666533.1</Id>
    <Id>NP_000552.1</Id>
  </IdList>
  <TranslationSet><Translation><From>human[organism]</From><To>"Homo sapiens"[Organism]</To></Translation></TranslationSet>
  <TranslationStack><TermSet><Term>gstm[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm1[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm1/t1[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm1a[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm1b[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm2[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm3[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm3a[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm3b[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm3c[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm3d[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm4[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm4'[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm4a[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm4b[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm4c[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm4d[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm5[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm6[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm7[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm8[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet><Term>gstm9[All Fields]</Term><Field>All Fields</Field><Count>43</Count></TermSet></TermSet></TranslationStack>
</eSearchResult>
urllib2/urlopen at the NCBI

#!/bin/env python
from urllib2 import urlopen
import re
# import pdb; pdb.set_trace()

# setup URL

s_args = "db=protein&term=GSTM*+AND+human[organism]+\n"+"+AND+srcdb_refseq[prop]&idtype=acc&retmax=1000";

acc_lines = urlopen(s_url+s_args).readlines() # get results

acc_re = re.compile(r'<Id>(\w+.\w+)</Id>') # setup RE to get ID's

acc_ids = [ m.group(1) for l in acc_lines for m in acc_re.search(l) if m ]

#for id in acc_ids:
#
# now we have a list of acc's, get the sequences
seq_args = "db=protein&id=+\",\".\",\'.join(acc_ids) + "+&rettype=fasta"

seq_html = urlopen(seq_url + seq_args).read() # get results

print seq_html

NCBI &retmode, &rettype


<table>
<thead>
<tr>
<th>Record Type</th>
<th>&amp;rettype</th>
<th>&amp;retmode</th>
</tr>
</thead>
<tbody>
<tr>
<td>text ASN.1</td>
<td>null</td>
<td>db = gene</td>
</tr>
<tr>
<td>XML</td>
<td>null</td>
<td>asn.1, default</td>
</tr>
<tr>
<td>Gene table</td>
<td>gene_table</td>
<td>xml</td>
</tr>
<tr>
<td>text ASN.1</td>
<td>null</td>
<td>db = nuccore, nucest, nucgss, protein or popset</td>
</tr>
<tr>
<td>binary ASN.1</td>
<td>null</td>
<td>text, default</td>
</tr>
<tr>
<td>Full record in XML</td>
<td>native</td>
<td>xml</td>
</tr>
<tr>
<td>FASTA</td>
<td>fasta</td>
<td>text</td>
</tr>
<tr>
<td>Accession</td>
<td>acc</td>
<td>text</td>
</tr>
<tr>
<td>text ASN.1</td>
<td>null</td>
<td>db = pubmed</td>
</tr>
<tr>
<td>XML</td>
<td>null</td>
<td>asn.1, default</td>
</tr>
<tr>
<td>MEDLINE</td>
<td>medline</td>
<td>xml</td>
</tr>
<tr>
<td>PMID list</td>
<td>uilist</td>
<td>text</td>
</tr>
<tr>
<td>Abstract</td>
<td>abstract</td>
<td>text</td>
</tr>
<tr>
<td>XML</td>
<td>null</td>
<td>db = taxonomy</td>
</tr>
<tr>
<td>TaxID list</td>
<td>uilist</td>
<td>xml, default</td>
</tr>
</tbody>
</table>

fasta.bioch.virginia.edu/biol4230
Introduction

The EMBL-EBI provides programmatic access to various data resources and analysis tools via Web Services technologies.

Web Services is an integration and inter-operation technology, to ensure client and server software from various sources will work well together, the technology is built on open standards:

- Representational state transfer (REST): a software architecture style.
- Web Services Description Language (WSDL): a method for describing Web Services and their capabilities.

For the transport layer Web Services utilise common network protocols, generally the HyperText Transfer Protocol (HTTP).

For an overview of Web Services technologies and short tutorials on using common programming languages and Web Services tool-kits see Introduction to Web Services.

<table>
<thead>
<tr>
<th>Service</th>
<th>Clients</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayExpress</td>
<td></td>
<td>Microarray data searching with ArrayExpress.</td>
</tr>
<tr>
<td>CHEBI Web Services</td>
<td>CHEBI Web Services</td>
<td>Entry retrieval from the CHEBI database.</td>
</tr>
<tr>
<td>CHEMBL Web Services</td>
<td>CHEMBL Web Services</td>
<td>Search data in, and retrieve data from the CHEMBL database</td>
</tr>
<tr>
<td>EBI-Eye</td>
<td>EBI-eye</td>
<td>Database search using the EBI-eye search engine.</td>
</tr>
<tr>
<td>ENA Browser</td>
<td></td>
<td>Retrieval of sequence and associated records from ENA</td>
</tr>
<tr>
<td>Gene Expression Atlas API</td>
<td></td>
<td>Enriched database of summary statistics over a curated subset of ArrayExpress Archive</td>
</tr>
<tr>
<td>MirtService</td>
<td></td>
<td>Database search and data retrieval using mirt.</td>
</tr>
<tr>
<td>PSIQUIC</td>
<td></td>
<td>Standardised access to molecular interaction databases, including CHEMBL, Reactions and IMAC</td>
</tr>
<tr>
<td>Rhea</td>
<td></td>
<td>Manually annotated database of chemical reactions</td>
</tr>
<tr>
<td>SRS</td>
<td></td>
<td>Database search and data retrieval using SRS@EBI.</td>
</tr>
<tr>
<td>UniProt.org</td>
<td></td>
<td>The Universal Protein Resource (UniProt) a comprehensive resource for protein sequence and annotation data.</td>
</tr>
<tr>
<td>WIDOTech (REST)</td>
<td>WIDOTech (REST)</td>
<td>Identifier based entry retrieval for various up-to-date biological databases.</td>
</tr>
<tr>
<td>WIDOTech (SOAP)</td>
<td>WIDOTech (SOAP)</td>
<td>Identifier based entry retrieval for various up-to-date biological databases.</td>
</tr>
</tbody>
</table>
How to find data: EBI/EMBL

www.ebi.ac.uk/Tools/webservices/

<table>
<thead>
<tr>
<th>REST Service</th>
<th>SOAP Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTA (REST)</td>
<td>FASTA (SOAP)</td>
<td>Fast protein or nucleotide comparison using the FASTA suite. Includes Smith and Waterman local-local (SSSEARCH), global-local (GLSEARCH) and global-global (GGSEARCH) alignment searches.</td>
</tr>
<tr>
<td>FASTM (REST)</td>
<td>FASTM (SOAP)</td>
<td>Peptide fragment searches using the FASTF, FASTM or FASTS programs from the FASTA suite.</td>
</tr>
<tr>
<td>NCBI BLAST (REST)</td>
<td>NCBI BLAST (SOAP)</td>
<td>Compare a sequence with those contained in nucleotide and protein databases using NCBI BLAST.</td>
</tr>
<tr>
<td>PSI-BLAST (REST)</td>
<td>PSI-BLAST (SOAP)</td>
<td>Position Specific Iterative BLAST (PSI-BLAST), guided mode.</td>
</tr>
<tr>
<td>PSI-Search (REST)</td>
<td>PSI-Search (SOAP)</td>
<td>Iterative Smith and Waterman using a PSI-BLAST strategy.</td>
</tr>
<tr>
<td>WU-BLAST (REST)</td>
<td>WU-BLAST (SOAP)</td>
<td>Compare a novel sequence with those contained in nucleotide and protein databases using WU-BLAST.</td>
</tr>
</tbody>
</table>

From alignments to domains, How to get from RefSeq to Pfam?

• Pfam uses Uniprot Id’s and Uniprot accession numbers:

<table>
<thead>
<tr>
<th>acc</th>
<th>id</th>
</tr>
</thead>
</table>
  >sp|043708|MAAI_HUMAN Maleylacetoacetate isomerase GH=GST21 PE=1|
  SV=3|MQAGKP|LYSYFRRSCVWVR1AALKKSIDYTVPIMLKD80Q9Q3KDFQALNPQKQPPLKIDGIT18Q6SLA|
  I2EYLDKRTQPLQDPKKAASVHR1MILLAGQ1QPQNLVLKQVOQ2H2QCNNAQH1TYOFH1LEILQ7YQ6W|
  YCVGDVMTADLCVPLVQVAAERFVSDLTPPTISSINHKLVLLEAPQVQVFCRCQFP0TP1E6

• UniProt provides a utility for mapping from other accession numbers to UniProt accessions/ids

  http://www.uniprot.org/faq/28#id_mapping_examples
# Mapping to/from UniProt accessions

http://www.uniprot.org/faq/28#id_mapping_examples

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbreviation</th>
<th>Direction</th>
<th>Name</th>
<th>Abbreviation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniProtKB AC/ID</td>
<td>ACC+ID</td>
<td>from</td>
<td>DNA</td>
<td>EMBL_ID</td>
<td>both</td>
</tr>
<tr>
<td>UniProtKB AC</td>
<td>ACC</td>
<td>to</td>
<td>DNA CDS</td>
<td>EMBL</td>
<td>both</td>
</tr>
<tr>
<td>UniProtKB ID</td>
<td>ID</td>
<td>to</td>
<td>PIR</td>
<td>PIR</td>
<td>both</td>
</tr>
<tr>
<td>UniParc UPARC</td>
<td>both</td>
<td></td>
<td>UniGene</td>
<td>UNIGENE_ID</td>
<td>both</td>
</tr>
<tr>
<td>UniRef50 NF50</td>
<td>both</td>
<td></td>
<td>Entrez Gene</td>
<td>P_ENTREZGENEID</td>
<td>both</td>
</tr>
<tr>
<td>UniRef90 NF90</td>
<td>both</td>
<td></td>
<td>GI number*</td>
<td>P_GI</td>
<td>both</td>
</tr>
<tr>
<td>UniRef100 NF100</td>
<td>both</td>
<td></td>
<td>IPI</td>
<td>P_IPI</td>
<td>both</td>
</tr>
</tbody>
</table>

## Other sequence databases

- DNA EMBL ID both
- DNA CDS EMBL both
- PIR PIR both
- UniGene UNIGENE_ID both
- Entrez Gene P_ENTREZGENEID both
- GI number* P_GI both
- IPI P_IPI both
- RefSeq P_REFSEQ_AC both

## 3D structure databases

- PDB PDB_ID both
- DisProt DISPROT_ID both
- HSSP HSSP_ID both

---

Mapping to/from UniProt

```python
import urllib, urllib2
import fileinput

url = 'http://www.uniprot.org/mapping/'

# build list of RefSeq protein accessions
refseq_list = []
for acc in fileinput.input():
    acc = acc.strip()
    refseq_list.append(acc)

# tell uniprot mapper what to do
params = {
    'to':'ACC',
    'from':'P_REFSEQ_AC',
    'format':'tab',
    'query':''.join(refseq_list)
}

# params MUST be encoded
data = urllib.urlencode(params)
request = urllib2.Request(url, data)
contact = "wrp@virginia.edu" # set your email address
request.add_header('User-Agent', 'Python %s' % contact)
response = urllib2.urlopen(request)

page = response.read(200000)

print page # note that response includes 'From:', 'To:'
# from_to_lines = page.split('n') # gives individual lines

http://www.uniprot.org/help/programmatic_access#id_mapping_examples
```

---
Mapping to/from UniProt

```bash
sh uniprot_map.py ref_seq.list

From  To
NP_001504.2  043708
Np_001504.2  A6NNB8
NP_714543.1  Q7RTV2
NP_665877.1  043708
NP_001503.1  Q53YD7
NP_001503.1  Q6F4G1
NP_001396.1  P26641
NP_001395.1  O15217
NP_671488.1  Q8NE79
NP_665683.1  P08263
NP_665683.1  Q5Z2C1
NP_004271.1  O43324
NP_000844.2  P30711
NP_000845.1  P30712
NP_000838.3  Q16772
NP_006294.2  Q13155
```

What about Pfam?

[pfam.xfam.org/help#tabview=tab8](http://pfam.xfam.org/help#tabview=tab8)

---

**RESTful Interface**

This is an introduction to the `RESTful` interface to the Pfam website. `REST` (or Representation State Transfer) refers to a style of building websites which makes it easy to interact programmatically with the services provided by the site. A programmatic interface, commonly called an Application Programming Interface (API) allows users to write scripts or programs to access data, rather than having to rely on a browser to view a site.

**Basic concepts**

**URLs**

A RESTful service typically sends and receives data over HTTP, the same protocol that’s used by websites and browsers. As such, the services provided through a RESTful interface are identified using URLs.

In the Pfam website we use the same basic URL to provide both the standard HTML representation of Pfam data and the alternative XML representation. To see the data for a particular Pfam-A family, you would visit the following URL in your browser:

http://pfam.sanger.ac.uk/family/Pfam-A

To retrieve the data in XML format, just add an extra parameter, `format=xml`, to the URL:

http://pfam.sanger.ac.uk/family/Pfam-A?format=xml

The response from the server will now be an XML document, rather than an HTML page.

back to top
What about Pfam?
pfam.xfam.org/help#tabview=tab8

#!/usr/bin/python
from urllib2 import urlopen
import sys
loc="http://pfam.xfam.org/
prot_url = "protein?entry="
fam_url="family?entry="
url = prot_url
xml = "/output=xml"

for acc in sys.argv[1:] :
    #print "====",loc+url+acc
    print urlopen(loc+url+acc).read()
Pfam protein annotation: (the web page)

Summary

GSTT1_HUMAN
This is the summary of UniProt entry GSTT1_HUMAN (P30711).

- **Description:** Glutathione S-transferase theta-1 (EC:2.5.1.18)
- **Source organism:** *Homo sapiens (Human)* (NCBI taxonomy ID: 9606)
- **Protein length:** 240 amino acids

Please note: when we update each new Pfam database release, we take a copy of the UniProt sequence database. This snapshot of UniProt forms the basis of the overview that you see here. It is important to note that, although some UniProt entries may be removed after a Pfam release, these entries will not be removed from Pfam until the next Pfam database release.

**Pfam domains**

This image shows the arrangement of the Pfam domains that we found on this sequence. Clicking on a domain will take you to the page describing that Pfam entry. The table below gives the domain boundaries for each of the domains.

<table>
<thead>
<tr>
<th>Source</th>
<th>Domain</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pfam A</td>
<td>GST N</td>
<td>6</td>
<td>76</td>
</tr>
<tr>
<td>Pfam A</td>
<td>GST C</td>
<td>111</td>
<td>201</td>
</tr>
</tbody>
</table>

What about Pfam?

```
#!/usr/bin/python

from urllib2 import urlopen
import sys

loc = "http://pfam.xfam.org/"
prot_url = "protein?entry=
"fam_url="family?entry=
url = prot_url
xml = "$output=xml"

for acc in sys.argv[1:] :  
#    print "****", loc+url+acc

print urlopen(loc+url+acc+xml).read()
```
Pfam protein annotation (xml):
curl 'http://pfam.xfam.org/protein/P09488?output=xml'

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pbm PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<pbm xmlns="http://pfam.sanger.ac.uk/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://pfam.sanger.ac.uk/
                       http://pfam.sanger.ac.uk/static/documents/schemas/pbm.xsd"
     release="24.0"
     release_date="2009-10-07">

  <entry entry_type="sequence" db="uniprot" db_release="57.6" accession="P30711"
         id="GSTT1_HUMAN">

    <description>
      <![[CDATA[
        Glutathione S-transferase theta-1 EC=2.5.1.18
      ]]>]</description>

    <taxonomy tax_id="9606" species_name="Homo sapiens (Human)"
             kingdom="Eukaryota"
             phylum="Metazoa"
             class="Chordata"
             subphylum="Craniata"
             superclass="Vertebrata"
             class="Euteleostomi"
             superorder="Mammalia"
             order="Eutheria"
             infraorder="Euarchontoglires"
             superfamily="Primates"
             superfamily="Haplorrhini"
             family="Catarrhini"
             subfamily="Hominidae"
             genus="Homo"/>

    <sequence length="240" md5="a9cdedfd8f1dce1b7d6c106be78cbc73"
              crc64="BD19F2BFDEF9F619"
              version="4">
      MGLELYLDLLSQPCRAVYIFAKKNDIPFELRIVDLIKGQHLSDAFAQVNPLKKVPALKDGDFTLTESVAILLYLTRKYKVPDYWYPQDLQARARVDEYLAWQHTTLRRSCLRALWHKVMFPVFLGEPVSPQTLAATLAELDVTLQLLEDKFLQNKAFLTGPHISLADLVAITELMHPVGAGCQVFEGRPKLATWRQRVEAAVGEDLFQEAHEVILKAKDFPPADPTIKQKLMPWVLAMIR
    </sequence>

    <matches>
      <match accession="PF02798" id="GST_N" type="Pfam-A">
        <location start="6" end="76" ali_start="17" ali_end="75" hmm_start="15" hmm_end="74"
                  evalue="4.2e-08" bitscore="42.20"/>
      </match>
      <match accession="PF00043" id="GST_C" type="Pfam-A">
        <location start="111" end="201" ali_start="119" ali_end="200" hmm_start="9" hmm_end="93"
                  evalue="0.00019" bitscore="30.30"/>
      </match>
    </matches>
  </entry>
</pbm>

Dealing with XML

- all we want to do is find:
Dealing with XML – xml.etree.ElementTree

#!/usr/bin/env python

import pdb; pdb.set_trace()
import xml.etree.ElementTree as ET
from urllib2 import urlopen
import sys
loc="http://pfam.xfam.org/
url = "protein?entry="
xml = "&output=xml"
for acc in sys.argv[1:] :
    pfam_xml = urlopen(loc+url+acc+xml).read()
    root = ET.fromstring(pfam_xml)
    matches = root.findall('.//{http://pfam.xfam.org/}matches')
    domains = {}
    if (matches is None): exit()  # possibly no matches
    for child in matches.getchildren():
        for child2 in child:
            domains.update(child.attrib)  # .update adds a dict
            domains.update(child2.attrib)

    print domains['id'],domains['start'],domains['end']
Dealing with XML – ElementTree

```python
import xml.etree.ElementTree as ET
from urllib2 import urlopen
import sys

url = "http://pfam.xfam.org/protein?entry="
xml = "&output=xml"
for acc in sys.argv[1:] :
    pfam_xml = urlopen(url+acc+xml).read()
    root = ET.fromstring(pfam_xml)
    matches = root.find('http://pfam.xfam.org/matches')
    if (matches is None): continue  # sometimes no matches
    domains = {}
    for child in matches:
        for child2 in child:
            domains.update(child.attrib)
            domains.update(child2.attrib)
            print domains['id'], domains['start'], domains['end']
```

pfam_xml.py gstd1_human
GST_N 6 76
GST_C 111 201
Pfam Family XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- information on Pfam-A family PF02798 (GST_N), generated: 13:36:46 12-Feb-2015 -->
release="27.0" release_date="2013-03-06">
<entry entry_type="Pfam-A" accession="PF02798" id="GST_N">
<description><![CDATA[
Glutathione S-transferase, N-terminal domain
]]></description>
<comment><![CDATA[
Function: conjugation of reduced glutathione to a variety of targets. Also included in the alignment, but are not GSTs: * S-cryrtallins from squid. Similarity to GST previously noted. * Eukaryotic elongation factors 1-gamma. Not known to have GST activity; similarity not previously recognised. * HSP26 family of stress-related proteins, including auxin-regulated proteins in plants and starvation proteins in E. coli. Not known to have GST activity. Similarity not previously recognised. The glutathione molecule binds in a cleft between N and C-terminal domains - the catalytically important residues are proposed to reside in the N-terminal domain [1].]]>
</comment>
</entry>
</pfam>
```

Pfam Family XML (cont.)

```xml
<clan_membership clan_acc="CL0172" clan_id="Thioredoxin" />
<go_terms>
<category name="function"><term go_id="GO:0005515">protein binding</term></category>
</go_terms>
<curation_details>
<status>CHANGED</status>
<seed_source>Overington</seed_source>
<previous_id>gluts;</previouse_id>
<num_archs>61</num_archs>
<num_seqs><seed>53</seed><full>5748</full></num_seqs>
<num_species>1695</num_species>
<num_structures>674</num_structures>
<percentage_identity>24</percentage_identity>
<av_length>72.80</av_length>
<av_coverage>29.98</av_coverage>
</curation_details>
<hmm_details hmmer_version="3.0" model_version="15" model_length="76">
<build_commands>hmmbuild -o /dev/null HMM SEED</build_commands>
<search_commands>hmmscan -e 23193494 -E 1000 --cpu 4 HMM pfamseq</search_commands>
<cutoffs>
<gathering><sequence>20.4%</sequence><domain>20.4%</domain></gathering>
<trusted><sequence>20.4%</sequence><domain>20.4%</domain></trusted>
</cutoffs>
</hmm_details>
</entry>
</pfam>
```
Pfam family XML (xml.etree.ElementTree)

```python
import xml.etree.ElementTree as ET
from urllib2 import urlopen
import sys

loc = "http://pfam.xfam.org/"
prot_url = "protein/"
fam_url = "family/"
xml = "?output=xml"
url = fam_url

for acc in sys.argv[1:]:  # acc is a pfamA_acc, PF01234, not a uniprot_acc
    pfam_xml = urlopen(loc+url+acc+xml).read()
    root = ET.fromstring(pfam_xml)
    entry = root.find('.//{http://pfam.xfam.org/}entry')
    details = root.find('.//{http://pfam.xfam.org/}hmm_details')

    print entry.attrib['accession'], details.attrib['model_length']
```

One last XML: NCBI esummary for protein length

```xml
<?xml version="1.0" encoding="UTF-8"?>
<eSummaryResult>
  <DocSum>
    <Id>121735</Id>
    <Item Name="Caption" Type="String">P09488</Item>
    <Item Name="Title" Type="String">RecName: Full=Glutathione S-transferase Mu 1; AltName: Full=GST HB subunit 4; AltName: Full=GST class mu 1; AltName: Full=GSTM1-1a; AltName: Full=GSTM1-1b; AltName: Full=GSTM4</Item>
    <Item Name="Extra" Type="String">gi|121735|sp|P09488.3|GSTM1_HUMAN[121735]</Item>
    <Item Name="G1" Type="Integer">121735</Item>
    <Item Name="CreateDate" Type="String">1989/07/01</Item>
    <Item Name="UpdateDate" Type="String">2016/01/20</Item>
    <Item Name="Flags" Type="Integer">0</Item>
    <Item Name="TaxId" Type="Integer">9606</Item>
    <Item Name="Length" Type="Integer">218</Item>
    <Item Name="Status" Type="String">live</Item>
    <Item Name="ReplacedBy" Type="String"></Item>
    <Item Name="Comment" Type="String"><![CDATA[ ]]]></Item>
    <Item Name="AccessionVersion" Type="String">P09488.3</Item>
  </DocSum>
</eSummaryResult>
```
One last XML:
NCBI esummary for protein length

#!/bin/env python
import xml.etree.ElementTree as ET
from urllib2 import urlopen
import sys
prot_db_id = "db=protein&id="
def get_summary(acc):
    ncbi_xml = urlopen(loc+prot_db_id+acc).read()
    tree = ET.fromstring(ncbi_xml)
    items = {}
    for item in tree.iter(tag='Item'):
        # no {http://...} because no xmlns:
        items.update({item.attrib['Name']:item.text})
    return items
for acc in sys.argv[1:]:
    info = get_summary(acc)
    # for key in info.keys(): print key, info[key]
    print 'Length ('+acc+'): ',info['Length']

Another last XML:
NCBI esearch.fcgii accessions
<Count>41</Count><RetMax>40</RetMax><RetStart>0</RetStart>
<IDList>
  <Id>NP_001278092.1</Id>
  <Id>NP_001129490.1</Id>
  <Id>NP_758859.1</Id>
  <Id>NP_071900.2</Id>
  <Id>NP_000843.1</Id>
  ... stuff deleted
  <Id>NP_000842.2</Id>
  <Id>NP_000841.1</Id>
  <Id>NP_671489.1</Id>
  <Id>NP_714543.1</Id>
  <Id>NP_665513.1</Id>
  <Id>NP_005552.2</Id>
</IDList>
<TranslationSet>
<Translation><From>human[organism]</From><To>"Homo sapiens"[Organism]</To></Translation>
<Translation><TermSet><Term>gstm</Term><Field>All Fields</Field><Count>43</Count>
  ... stuff deleted
</Term></TranslationSet>
</eSearchResult>
Another last XML:
NCBI esearch.fcgi accessions

#!/bin/env python
import xml.etree.ElementTree as ET
from urllib2 import urlopen
import sys
search_string='GSTM*+AND+human[organism]+AND+srcdb_refseq[prop]'
def get_accs(search_str):
    prot_db_id='db=protein&idtype=acc&retmax=1000&term='
    ncbi_xml = urlopen(loc+prot_db_id+search_str).read()
    tree = ET.fromstring(ncbi_xml)
    acc_list = []
    for item in tree.iter(tag='Id'):
        acc_list.append(item.text)
    return acc_list
acc_list = get_accs(search_string)
for acc in acc_list:
    print acc

Bioinformatics Web Resources

- NCBI – eutilities: esearch/efetch/blast search
  www.ncbi.nlm.nih.gov/books/NBK25501/

- Recognizing web addresses (URLs)

- EBI – web services
  www.ebi.ac.uk/Tools/webservices/

- Uniprot ID mapper
  www.uniprot.org/faq/28#id_mapping_examples

- Pfam – using XML data
  pfam.xfam.org/help#tabview=tab10
  xml.etree.ElementTree
Homework, due Monday, Feb 20 (biol4230/hwk5)

Do the exercises and write the programs to answer the questions at:

fasta.bioch.virginia.edu/biol4230/labs/accessions_hwk5.html