Unison Tutorial

Reece Hart

2006-05-12
Tutorial Outline

- Introduction
  - data sources, algorithms, update scheme
- Schema
  - overview, design themes, critical tables
- Access
  - web pages, command line tools, perl API, psql
- Example Queries
  - Finding sequences
  - Finding parameters
  - Getting predictions for a sequence
  -Mining for sequence based on predictions
  - Tips
- Future Plans
What Can I Do With Unison?

- Retrieve sequence analysis for a single sequence.
- Mine for sequences based on predicted features, sequence origins, taxonomy, patents, orthology, and structure.
- Find all sources of a single sequence.
- Find patents for a sequence.
- Locate sequence variations relative to domains and in structure.
- Build new tools.
Design Goals

- Sequences are stored non-redundantly.
  - eliminates redundant computation and analysis
  - Results are keyed to sequences, parameters, and optionally a model.
  - Sequences are immutable and therefore results are never stale.
  - Sequences are linked to their origins and aliases.
- Fast, reliable, differential updates.
- Multiple result sets for different invocations
- Make no assumptions and provide no interpretations.
- Synopses of prediction results only, but and enable regeneration of results.
Unison Contents

- Non-redundant Sequences
  - UniProtKB/Swiss-Prot, IPI, Genengenes, Genehub representative sequences, RefSeq, Curagen, Incyte, ..., Ensembl *ab initio*, miscellaneous fragments

- Non-redundant Results
  - Pfam, TMHMM, SignalP, protcomp
  - BIG-PI, PSI-PRED, RegExp motifs
  - disprot, dispro, pmap

- Lots of other Data
  - patents, PDB, SCOP, GO, GOng, NCBI tax, HomoloGene, MINT, ...

- Statistics
  - 75 tables, 108 views, 120 functions
  - ~6 CPU-years' worth of data, >440M protein features
  - 14GB of compressed data, 130GB on disk w/indexes
**Update Procedure and Run Sets**

1. **Phase 1:** Load sequences and models (1 day)
2. **Phase 2:** Build sequence “run” sets (1/2 day)
3. **Phase 3:** Run and load (7 days?)
4. **Phase 4:** Mat’lized views (1/2 day)
5. **Phase 5:** Copy and push to production (web too) (1 day)

**Publicize** for external site (2 days)

---

**Set** | **Criteria** | **Algorithms**
--- | --- | ---
runA | human 100-1000 AA reliable origins | prospect antigenic BLAST†
runB | human, mouse, rat 100-1500 AA reliable origins | Pfam (fs & ls) BIG-PI RegExp PSIPRED
runC | human, mouse, rat, cow, zebrafish 100-3000 AA all sequence sources | pepcoil SignalP TMHMM antigenic pmap protcomp

*ad hoc* | | dispro†

†† these methods are currently unsupported

---

*data pushes to Oracle*
Implementation

- **Hardware**
  - hostname csb
  - 4 dual-core Opterons, 2.4 GHz
  - 32GB RAM
  - 500GB FC-RAID

- **Linux**
  - SuSE 10.0, kernel 2.6

- **PostgreSQL 8.1.3**
  - 3 databases: csb, csb-stage, csb-dev
  - unison is a schema within each

- **Perl 5.8**

- **Apache 2.0 web pages**
Unison Schema
Design Themes

• Abstraction and Normalization
  – most tables are essentially data types
  – expect a lot of joins, but views exist for common queries
  – facilitates updates of new params, etc

• Rely on database for correctness
  – pedantic and paranoid use of triggers and constraints

• Selective incorporation of external databases
  – schemas: unison, ncbi, tax, dali, go, pdb
Results Cube

feature types (HMM, TM, signal, etc)

Sequence Analysis
show structures predictions for a given sequence
computing these takes minutes-hours

Feature-Based Mining
show sequences which align to specified structures
computing these takes days-months

sequences

parameter slices
Critical Tables, Views, Functions

- Tables and views
  - porigin, pseq, pfptype, params
  - palias (view) and current_annotations_v
  - pahmm + pmhmm => pahmm_v
  - pfbigpi => pfbigpi_v
  - pfsignalpnn
  - psprotcomp + psprotcomp_locations => psprotcomp_v
  - pfregexp + pmregexp => pfregexp_v
  - run_history => run_history_v

- Functions
  - pseq_id_from_sequence()
  - params_id() and preferred_params_id_by_pftype()
  - porigin_id()
Table and Column Names

- **X_id** is always a primary or foreign key
  - foreign and primary keys always have the same name
  - except for pairwise sequence comparisons which use q_pseq_id and t_pseq_id
- **psX** – protein sequence property
  - e.g., psprotcomp
- **pfX** – protein feature
  - e.g., pfsignalpnn
- **paX** and **pmX** – protein alignments to models
  - e.g., pmhmm and pahmm
View Name Suffixes

- 
  - _mv – materialized view
  - _dv – “defining view”
    - mostly for internal use
    - not optimized, often slow
    - look for a corresponding _mv
  - _cv – “canned view”
    - views for complex data mining
    - exposed to public
    - see canned_views table
  - _v – ye olde standard view
Accessing Unison

• How:
  - native protocol
  - `psql` interactive shell (akin to `sqlplus`)
  - perl DBI
  - `perl API`
  - ODBC/JDBC
  - `web pages (and linking to them)`
  - `command line tools`
  - Oracle snapshot in biodev1 and bioprd1

• Details:
  - host `csb`
  - database `csb` or `csb-dev`
  - as self with Kerberos (user `PUBLIC`, no password is deprecated)

☞ Use Kerberos authentication wherever possible. Be prepared to change logins which use `PUBLIC`.
Web Page Linking

- **http://csb/unison/cgi/pseq_summary.pl?q=<query>**
  - query may be pseq_id, alias, md5, or sequence itself
  - all links use GET method (i.e., params in URL)

- **Sequence analysis pages**
  - summary page
  - aliases
  - patents
  - protein features
  - protein structure and variant mapping
  - HomoloGene homologs
  - pmap and blat genome maps

- **Other pages**
  - feature-based mining
  - browse “canned views”
unison-get-seq

- Fetch sequences by pseq_id or alias, from stdin or args
- Useful flags:
  - -A select by alias
  - -b "best alias" in defline
  - -B "best annotation" in defline
  - -u Unison:pseq_id in defline (-bu may be used together)
  - --iupac20 select only sequences with standard AA
  - -v verbose – prints progress to stderr
- Example:
  - kinit
  - unison-get-seq -Abu TNFA_HUMAN

>Unison:98 UniProtKB/Swiss-Prot:TNFA_HUMAN (Tumor necrosis factor precursor (TNF-alpha)
  (Tumor necrosis factor ligand superfamily member 2) (TNF-a) (Cachectin))
MSTESMIRDVELAEELPKKTGGPQGSRRCLFLSLFSFLIVAGATTLFCLLHFGVIGPQR
EEFPRDLSISPLAQAVRSSRTPSDKPVAVHVVANPQAEGQLQWLNRRANALLANGVELR
DNQLVVPSEGLYLIYSQVLFKGQGCPSTHVLLTHTISRIAHSVYOTKVNLLSAIKSPCQRE
TPEGAEAKPWYEPIYLGGVQLEKGDRLSAEINRPYLDFAESGQVYFGIIAL
unison-annotation

- Provides features and other information for sequences specified by pseq_id, alias, or fasta

```
$ unison-annotation -S <myseqs.fa
*Unison:98:UniProtKB/Swiss-Prot:TNFA_HUMAN (Tumor necrosis fac...
*signalp (SignalP 3.0 (euk), ran on 2005-11-03 16:40)
*tm (TMHMM 2.0c, ran on 2005-11-16 14:05)
#start stop 35 57
*regexp (regexp, ran on 2005-11-16 11:26)
#start stop feature 133 138 ITIM
*pfam (Pfam 19.0 ls, ran on 2006-01-13 21:30)
#start stop score eval acc feature descr
102 233 210 4.8e-60 PF00229.8 TNF TNF(Tumour Necrosis Factor) family
*protcomp (protcomp default, ran on 2005-08-10 05:40)
#loc Plasma membrane
*alia#alias origin descr
NP_000585 RefSeq tumor necrosis factor alpha [Homo sapiens].
PR034403 GenenGenes Human TNF-a
PR021907 GenenGenes Human TNF-a
PR06 GenenGenes Human TNF-a
IPI00001671.1 IPI Tumor necrosis factor precursor; SWISS-PROT:P01375|
```
Unison perl API

- Unison perl module
  - acts like a subclass of DBI
  - adds many utility methods
  - is stable but homely
- The module is available to all users of /usr/local/tools/bin/perl by default
- more info:
  - perldoc Unison
  - /gne/research/apps/unison/examples/

```perl
use Unison;
my $u = new Unison;
my $q = $u->pseq_id_by_sequence('YGGFM');
my $np = $u->selectrow_array("SELECT count(*) FROM patents_v WHERE pseq_id=?", undef, $q);
my $sth = $u->prepare('select * from run_history_v where pseq_id=?');
$sth->execute($q);
$sth->dump_results();
```
psql

PostgreSQL's command-line interface
psql intro

- Do this:
  - `kinit`
  - `psql -h csb -d csb`

- **The most important commands**
  - `\?` – help
  - `\d+ [selection]` – describe object
  - `\dv+ [selection]` – view summary
  - `\dt+ [selection]` – table summary
  - `\df+ [selection]` – function summary
  - selection may be:
    - a table, view, or function name, optionally schema-qualified
      - e.g., pseq, unison.palias
    - a shell-style glob
  - `explain [query]`
Browsing Tables and Views

```
unison@csb=> \dt+ unison.

<table>
<thead>
<tr>
<th>Schema</th>
<th>Name</th>
<th>Type</th>
<th>Owner</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unison</td>
<td>_readme</td>
<td>table</td>
<td>unison</td>
<td>READ THIS FIRST -- Unison</td>
</tr>
<tr>
<td>unison</td>
<td>best_annotation_mv</td>
<td>table</td>
<td>unison</td>
<td>materialized view of best</td>
</tr>
<tr>
<td>unison</td>
<td>canned_views</td>
<td>table</td>
<td>unison</td>
<td>curated data mining views</td>
</tr>
<tr>
<td>unison</td>
<td>ensembl_coordinates_mv</td>
<td>table</td>
<td>unison</td>
<td></td>
</tr>
<tr>
<td>unison</td>
<td>ensembl_unambiguous_coordinates_mv</td>
<td>table</td>
<td>unison</td>
<td></td>
</tr>
<tr>
<td>unison</td>
<td>ensembl_unambiguous_overlaps_mv</td>
<td>table</td>
<td>unison</td>
<td></td>
</tr>
<tr>
<td>unison</td>
<td>genasm</td>
<td>table</td>
<td>unison</td>
<td>genome and assembly</td>
</tr>
</tbody>
</table>
```

```
unison@csb=> \dv+ unison.*pmap*

<table>
<thead>
<tr>
<th>Schema</th>
<th>Name</th>
<th>Type</th>
<th>Owner</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unison</td>
<td>pmap_aln_unambiguous_overlaps_v</td>
<td>view</td>
<td>unison</td>
<td></td>
</tr>
<tr>
<td>unison</td>
<td>pmap_aln_unambiguous_v</td>
<td>view</td>
<td>unison</td>
<td></td>
</tr>
<tr>
<td>unison</td>
<td>pmap_v</td>
<td>view</td>
<td>unison</td>
<td>view of pmap alignments ...</td>
</tr>
</tbody>
</table>
```
unison@csb=> \d+ pahmm

Table "unison.pahmm"

<table>
<thead>
<tr>
<th>Column</th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pfeature_id</td>
<td>//</td>
<td>unique feature id</td>
</tr>
<tr>
<td>pseq_id</td>
<td>//</td>
<td>unique protein sequence identifier -- see pseq(pseq_id)</td>
</tr>
<tr>
<td>pftype_id</td>
<td>//</td>
<td>protein feature type identifier -- see pftype(pftype_id)</td>
</tr>
<tr>
<td>start</td>
<td>//</td>
<td>start of prediction in protein sequence</td>
</tr>
<tr>
<td>stop</td>
<td>//</td>
<td>stop of prediction in protein sequence</td>
</tr>
<tr>
<td>params_id</td>
<td>//</td>
<td>parameter set identifier -- see params(params_id)</td>
</tr>
<tr>
<td>pmodel_id</td>
<td>//</td>
<td>unique protein model identifier</td>
</tr>
<tr>
<td>mstart</td>
<td>//</td>
<td>start of match /in model/</td>
</tr>
<tr>
<td>mstop</td>
<td>//</td>
<td>stop of match /in model/</td>
</tr>
<tr>
<td>ident</td>
<td>//</td>
<td></td>
</tr>
<tr>
<td>sim</td>
<td>//</td>
<td></td>
</tr>
<tr>
<td>gaps</td>
<td>//</td>
<td></td>
</tr>
<tr>
<td>qgaps</td>
<td>//</td>
<td>number of gaps in query sequence</td>
</tr>
<tr>
<td>tgaps</td>
<td>//</td>
<td>number of gaps in target sequence</td>
</tr>
<tr>
<td>score</td>
<td>//</td>
<td>algorithm-specific score</td>
</tr>
<tr>
<td>eval</td>
<td>//</td>
<td>expectation value</td>
</tr>
<tr>
<td>len</td>
<td>//</td>
<td></td>
</tr>
</tbody>
</table>

Indexes:
- "pahmm_redundant_feature" UNIQUE, btree (pseq_id, "start", stop, pmodel_id, params_id, mstart, mstop) CLUSTER
- "pahmm_search1" btree (pmodel_id, eval, params_id)
- "pahmm_search2" btree (params_id, eval, pmodel_id)
- "pahmm_search3" btree (params_id, score, pmodel_id)

Foreign-key constraints:
- "pahmm_params_id_exists" FOREIGN KEY (params_id) REFERENCES params(params_id) ON UPDATE CASCADE ON DELETE CASCADE
### Viewing a View Definition

**View Definition: View "unison.pmap_v"**

Table:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Modifiers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params_id</td>
<td>integer</td>
<td></td>
<td>parameter set identifier -- see params</td>
</tr>
<tr>
<td>genasm_id</td>
<td>integer</td>
<td></td>
<td>genome assembly identifier -- see genasm</td>
</tr>
<tr>
<td>pseq_id</td>
<td>integer</td>
<td></td>
<td>unique protein sequence identifier -- see pseq</td>
</tr>
<tr>
<td>aln_id</td>
<td>integer</td>
<td></td>
<td>pmap_aln alignment identifier</td>
</tr>
<tr>
<td>pstart</td>
<td>integer</td>
<td></td>
<td>start of alignment in protein sequence</td>
</tr>
<tr>
<td>pstop</td>
<td>integer</td>
<td></td>
<td>stop of alignment in protein sequence</td>
</tr>
<tr>
<td>exons</td>
<td>bigint</td>
<td></td>
<td>number of exons</td>
</tr>
<tr>
<td>aln_length</td>
<td>bigint</td>
<td></td>
<td>length of alignment</td>
</tr>
<tr>
<td>pct_cov</td>
<td>integer</td>
<td></td>
<td>percent coverage</td>
</tr>
<tr>
<td>ident</td>
<td>integer</td>
<td></td>
<td>percent identity</td>
</tr>
<tr>
<td>chr</td>
<td>text</td>
<td></td>
<td>chromosome</td>
</tr>
<tr>
<td>strand</td>
<td>character(1)</td>
<td></td>
<td>genomic strand ('+' or '-')</td>
</tr>
<tr>
<td>gstart</td>
<td>integer</td>
<td></td>
<td>genomic start position on chromosome</td>
</tr>
<tr>
<td>gstop</td>
<td>integer</td>
<td></td>
<td>genomic stop position on chromosome</td>
</tr>
</tbody>
</table>

**View definition:**

```sql
SELECT a.params_id, a.genasm_id, h.pseq_id, ah.aln_id, min(h.pstart) AS pstart, ...
FROM pmap_hsp h
JOIN pmap_alnhsp ah ON h.hsp_id = ah.hsp_id
JOIN pmap_aln a ON ah.aln_id = a.aln_id
JOIN pseq q ON h.pseq_id = q.pseq_id
GROUP BY a.params_id, a.genasm_id, h.pseq_id, ah.aln_id, h.chr, h.strand, ...
ORDER BY h.pseq_id, (sum(h.pstop - h.pstart + 1)::double precision / ...
Reality Queries

(using psql, but applies to all access modes)
Finding A Sequence

- By sequence
  - `select pseq_id_from_sequence('MYSEQ')`

- By md5
  - `select pseq_id from pseq where md5='f9e8d7...';`

- By alias
  - `select pseq_id from palias where alias='PR054321'`
  - `or alias LIKE 'NP_00123%'`
  - `or alias ~ '^NP_0123'`
  - `AND porigin_id=porigin_id('RefSeq')`
  - `AND porigin_id=porigin_id('GenenGenes')`
  - `AND tax_id=gs2tax_id('BRARE')`

- By NCBI gene
  - `select pseq_id from palias where porigin_id=porigin_id('RefSeq gi') and alias=73967277;`
Finding Parameters

- select params_id,name,descr from params;

<table>
<thead>
<tr>
<th>params_id</th>
<th>name</th>
<th>descr</th>
<th>commandline</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>BIG-PI default</td>
<td>BIG-PI GPI prediction; <a href="http://bigpi">http://bigpi</a> metazoa %s short</td>
<td>bigpi metazoa %s short</td>
</tr>
<tr>
<td>3</td>
<td>BLAST</td>
<td>UNSUPPORTED! Unison internal u</td>
<td>blastall -FF -z2829482 -p blas</td>
</tr>
<tr>
<td>36</td>
<td>Bcl-2 ls</td>
<td>Bcl-2 homology domains custom</td>
<td>ldhmmpfam --acc -E10 -Z8183</td>
</tr>
<tr>
<td>4</td>
<td>EMBOSS/antigenic</td>
<td>antigenicity predictions; http</td>
<td>antigenic -minlen 6 -rformat s</td>
</tr>
<tr>
<td>37</td>
<td>EMBOSS/pepcoil</td>
<td>EMBOSS pepcoil coiled-coil pre</td>
<td>pepcoil -noother -window 28 -f</td>
</tr>
<tr>
<td>5</td>
<td>EMBOSS/sigcleave</td>
<td>signal cleavage prediction; ht</td>
<td>sigcleave -minweight 3.5 -rfor</td>
</tr>
<tr>
<td>11</td>
<td>Genome BLAT</td>
<td>genomic localization of protein</td>
<td>gfClient -t=dnax -q=prot trp 1</td>
</tr>
<tr>
<td>38</td>
<td>PMAP 2006-03-20</td>
<td>genomic localization of protein</td>
<td>pmap.2006-03-20 -d NHGD_R35 -B</td>
</tr>
<tr>
<td>8</td>
<td>PSSM default</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Pfam 19.0 fs</td>
<td>Pfam 19.0 /f/ragmentary models</td>
<td>ldhmmpfam --acc -E10 -Z8183</td>
</tr>
<tr>
<td>34</td>
<td>Pfam 19.0 ls</td>
<td>Pfam 19.0 local models</td>
<td>ldhmmpfam --acc -E10 -Z8183</td>
</tr>
<tr>
<td>14</td>
<td>Prospect2 SCOP default</td>
<td>Prospect protein threading; ht</td>
<td>-scop</td>
</tr>
<tr>
<td>1</td>
<td>Prospect2 default</td>
<td>Prospect protein threading; ht</td>
<td>-global</td>
</tr>
<tr>
<td>2</td>
<td>Prospect2 global_local</td>
<td>Prospect protein threading; ht</td>
<td>-global_local</td>
</tr>
<tr>
<td>23</td>
<td>Prospect2 ssp_psipred default</td>
<td>Prospect protein threading; ht</td>
<td>-global</td>
</tr>
<tr>
<td>17</td>
<td>Psipred v2.45</td>
<td>PSIPRED secondary structure pr</td>
<td>runpsipred -j 3 -h 0.001 -a 2</td>
</tr>
<tr>
<td>28</td>
<td>SignalP 3.0 (euk)</td>
<td>Signal sequence prediction per</td>
<td>/gne/compbio/i686-linux-2.6/bi</td>
</tr>
<tr>
<td>29</td>
<td>TMHMM 2.0c</td>
<td>TMHMM 2.0c, <a href="http://www.cbs.dtu">http://www.cbs.dtu</a></td>
<td>/gne/compbio/i686-linux-2.6/op</td>
</tr>
<tr>
<td>41</td>
<td>dispro</td>
<td><a href="http://www.ics.uci.edu/~baldig">http://www.ics.uci.edu/~baldig</a></td>
<td>MANUAL</td>
</tr>
<tr>
<td>39</td>
<td>disprot VL3H</td>
<td>disprot protein disorder predi</td>
<td>MANUAL: temple-disprot.pl</td>
</tr>
<tr>
<td>31</td>
<td>gne_prospect</td>
<td>Prospect protein threading; ht</td>
<td>gne_prospect -a 2 -p -b</td>
</tr>
<tr>
<td>0</td>
<td>no args</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>protcomp default</td>
<td>ProtComp protein localization</td>
<td>protcomp</td>
</tr>
<tr>
<td>12</td>
<td>regexp</td>
<td>Genentech in-house TM detection</td>
<td>tmdetect</td>
</tr>
<tr>
<td>9</td>
<td>tmdetect default</td>
<td>Genentech in-house TM detection</td>
<td>tmdetect</td>
</tr>
</tbody>
</table>

- commandline in params
- Consider using:
  - select params_id('Pfam 19.0 ls')
  - select preferred_params_id_by_pftype('HMM')
What's Been Run?

- **Always** consult `run_history_v` to see whether Unison contains the results you seek.

```sql
unison@csb=> select * from run_history_v where pseq_id=76;
```

<table>
<thead>
<tr>
<th>pseq_id</th>
<th>params_id</th>
<th>params</th>
<th>ran_on</th>
<th>failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>19</td>
<td>BIG-PI default</td>
<td>2005-02-06 01:50</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>3</td>
<td>BLAST</td>
<td>2003-08-18 14:36</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>36</td>
<td>Bcl-2 ls</td>
<td>2006-02-01 19:50</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>4</td>
<td>EMBOSS/antigenic</td>
<td>2003-10-07 20:56</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>37</td>
<td>EMBOSS/pepcoil</td>
<td>2006-05-03 18:37</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>5</td>
<td>EMBOSS/sigcleave</td>
<td>2003-10-13 16:30</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>11</td>
<td>Genome BLAT</td>
<td>2004-02-11 10:48</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>38</td>
<td>PMAP 2006-03-20</td>
<td>2006-03-23 14:53</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>33</td>
<td>Pfam 19.0 fs</td>
<td>2006-01-12 03:05</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>34</td>
<td>Pfam 19.0 ls</td>
<td>2006-01-13 21:20</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>1</td>
<td>Prospect2 default</td>
<td>2004-02-25 15:14</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>1</td>
<td>Prospect2 default</td>
<td>2004-02-20 08:03</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>17</td>
<td>Psipred v2.45</td>
<td>2005-06-11 07:51</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>28</td>
<td>SignalP 3.0 (euk)</td>
<td>2005-11-03 16:22</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>29</td>
<td>TMHMM 2.0c</td>
<td>2005-11-16 08:34</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>41</td>
<td>dispro</td>
<td>2006-05-04 22:42</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>39</td>
<td>disprot VL3H</td>
<td>2006-05-04 17:42</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>20</td>
<td>protcomp default</td>
<td>2005-08-10 03:11</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>12</td>
<td>regexp</td>
<td>2005-11-16 11:26</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>12</td>
<td>regexp</td>
<td>2006-04-13 04:47</td>
<td>f</td>
</tr>
<tr>
<td>76</td>
<td>9</td>
<td>tmdetect default</td>
<td>2006-02-21 17:38</td>
<td>f</td>
</tr>
</tbody>
</table>
Fetching Pfam Domains

- The hard way

```
unison@csb-dev=> SELECT A.pseq_id,M.name,M.acc,A.eval,A.score,M.descr
FROM pahmm A
JOIN pmhmm M on A.pmodel_id=M.pmodel_id
WHERE A.params_id=34 AND A.eval<1e-10 AND pseq_id=15;
```

<table>
<thead>
<tr>
<th>pseq_id</th>
<th>name</th>
<th>acc</th>
<th>eval</th>
<th>score</th>
<th>descr</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Insulin</td>
<td>PF00049.8</td>
<td>6e-23</td>
<td>86</td>
<td>Insulin/IGF/Relaxin family</td>
</tr>
<tr>
<td>15</td>
<td>IGF2_C</td>
<td>PF08365.1</td>
<td>2.2e-35</td>
<td>128</td>
<td>Insulin-like growth factor II E-</td>
</tr>
</tbody>
</table>

(2 rows)

Time: 1.900 ms
Fetching Pfam Domains 2

- A better way
  - use the pahmm_v view
  - for queries you want to rerun, use preferred_params_id_by_pftype

```
unison@csb-dev=> SELECT pseq_id,name,acc,eval,score,descr
     FROM pahmm_v
    WHERE params_id=preferred_params_id_by_pftype('HMM') AND eval<1e-10 AND pseq_id=15;
```

```
pseq_id | name   | acc    | eval   | score | ends | descr
---------+---------+--------+--------+-------+------+----------------------------------
   15 | IGF2_C  | PF08365.1 | 2.2e-35 |  128  |  []  | Insulin-like growth factor...
   15 | Insulin | PF00049.8 |  6e-23  |    86 |  []  | Insulin/IGF/Relaxin family...
```

(2 rows)

Time: 2.574 ms
Fetching “All” Features

- `pseq_features_v` provides *current* results for *common* feature

```sql
unison@csb-dev=> SELECT feature,start,stop,score,eval,descr FROM pseq_features_v WHERE pseq_id=82;

<table>
<thead>
<tr>
<th>feature</th>
<th>start</th>
<th>stop</th>
<th>score</th>
<th>eval</th>
<th>descr</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>1</td>
<td>31</td>
<td>0.884</td>
<td></td>
<td>signal sequence</td>
</tr>
<tr>
<td>TM</td>
<td>7</td>
<td>29</td>
<td></td>
<td></td>
<td>transmembrane domain</td>
</tr>
<tr>
<td>PAN_1</td>
<td>34</td>
<td>124</td>
<td>64</td>
<td>3.9e-16</td>
<td>PAN domain</td>
</tr>
<tr>
<td>Kringle</td>
<td>128</td>
<td>206</td>
<td>178</td>
<td>2e-50</td>
<td>Kringle domain</td>
</tr>
<tr>
<td>Kringle</td>
<td>211</td>
<td>288</td>
<td>178</td>
<td>1.6e-50</td>
<td>Kringle domain</td>
</tr>
<tr>
<td>Kringle</td>
<td>305</td>
<td>383</td>
<td>181</td>
<td>2e-51</td>
<td>Kringle domain</td>
</tr>
<tr>
<td>Kringle</td>
<td>391</td>
<td>469</td>
<td>155</td>
<td>1.3e-43</td>
<td>Kringle domain</td>
</tr>
<tr>
<td>Trypsin</td>
<td>495</td>
<td>716</td>
<td>223</td>
<td>4.3e-64</td>
<td>Trypsin</td>
</tr>
<tr>
<td>Y-ITIM</td>
<td>619</td>
<td>628</td>
<td>223</td>
<td>4.3e-64</td>
<td>ITIM motif with second upstream Y; more ...</td>
</tr>
<tr>
<td>ITIM</td>
<td>623</td>
<td>628</td>
<td></td>
<td></td>
<td>Immunotyrosine Inhibition Motif</td>
</tr>
</tbody>
</table>

(10 rows)

Time: 8.624 ms
Wrap Up
Dos and Don'ts
(Not “Dos and Donuts”)

• Do
  - specify params_id in all queries for results.
  - consult the run_history_v view, especially when your query returns no results.
  - use the helper functions params_id, params_id_by_pftype, porigin_id, etc.
  - experiment with Unison.
  - provide feedback and suggestions.

• Don't
  - hesitate to ask questions.
  - rely on PUBLIC user.
Developer Resources

- **Unison**
  - [http://csb/unison/tour/](http://csb/unison/tour/)
  - [/gne/research/apps/unison/examples/](http://csb/unison/tour/)
  - `perldoc Unison`

- **PostgreSQL**
  - `psql \d commands`
  - [http://www.postgresql.org/docs/](http://www.postgresql.org/docs/)

- **cvs** -d :ext:csb/srv/cvs unison
  - contains code examples

- **cvs** -d :ext:csb/srv/cvs unison-web
The Future

- Kiran will assume a greater role for Unison
  - nearly all data loading and maintenance
  - user help
  - API maintenance
  - Oracle data pushes

- I will continue to be involved in
  - mining efforts using Unison
  - added new data types
  - schema, API, and web oversight