Programmatic Access to Ensembl Genomes

Paul Kersey

The Ensembl Web Browser is the face of the system, but...
- Not so suitable for bulk data exploration
- Very slow
- Fortunately, there are a variety of alternative approaches for accessing the data programmatically (i.e. automatically)
- Mostly these utilise standard abstractions for data representation, in order to take advantage of standard tools designed around these abstractions
- Generality = code re-usability

Ensembl
- A modular suite of software for genome analysis and visualisation developed jointly by the Wellcome Trust Sanger Institute and the European Bioinformatics Institute
- Now used for genomes from across the taxonomic space
- Offers a standard set of interfaces to a wide range of genome-scale data, including:
  - Web-based GUI
  - Public mySQL server
  - Perl and RESTful API
  - FTP
  - Data mining tool (constructed using BioMart) framework with its own set of interfaces: web GUI, web services, command line and local client
Access method 1: FTP downloads

- http://plants.ensembl.org/info/data/ftp/index.html
- Genomic, cDNA and protein sequence (FASTA)
- Annotated sequence (EMBL / GenBank)
- Gene sets (GTF)
- Resequencing alignments individuals / strains (EMF)
- Whole-genome multiple alignments (EMF)
- Gene-based multiple alignments (EMF)
- Constrained elements (BED)
- Database dumps (MySQL)

Access method 2: MySQL

- MySQL: an open-source relational database management system (RDBMS)
- Used as the back end to support most Ensembl pipelines and applications
- You get the database from http://mysql.com and install locally
- On the Ensembl Genomes FTP site, you can download the Ensembl schema as a .sql file.
- You can also download the data files

```
/data/mysql/bin/mysql -u mySQLdb
create database zea_mays_core_24_77_6;
exit;
/data/mysql/bin/mysql -u mySQLdb zea_mays_core_24_77_6 < zea_mays_core_24_77_6.sql
/data/mysql/bin/mysqlimport --fields_escaped_by =\ \ zea_mays_core_24_77_6 -L *.txt
```

Relational Database Management Systems (RDBMS)

- Relational database – define data structure as a specific instance (schema) of a general model
- RDBMS supports
  - Data definition according to a relational model
  - Impostion of integrity constraints and performance optimisations on the model
  - Data access via a standard grammar – Structured Query Language (SQL)

What does the Ensembl schema look like?

- Divided into separate schemas for different data types
  - core (sequence and gene models)
  - variation
  - compara (comparative genomics)
  - other_features
  - regulation
- Up to one instance of each type for each species
  - create database zea_mays_core_24_77_6
  - Certain tables/identifiers are shared between schemas to support integrated querying

And if you don’t want to run a mysql server yourself...

- Public access server at mysql -hmysql-esg-publicsql.ebi.ac.uk -uanonymous -p12345
  All non-vertebrate species
- Similar servers for vertebrates available at
  mysql -hensembldb.ensembl.org -uanonymous -p12345
  mysql -hmartdb.ensembl.org -uanonymous -p12345

What does the Ensembl schema look like?

- Schema is quite highly normalised, and based around the central dogma
- Some keys (e.g. gene_id) posted redundantly to reduce the number of table joins required
- Separation of internal and externally facing attributes
  - gene_d/orus gene_stable_id
  - db_name vs db_display_name
- Some m:n linkages
  - exon--exon_transcript--transcript
  - X: object_xref – xref – external_db
  - xrefs to many Ensembl object types can be stored in the one table
  - X:enum('RawContig','Transcript', 'Gene', 'Translation', 'Operon', 'OperonTranscript', 'Marker')
Ensembl Variation Database schema

MySQL example

Query to retrieve the Ensembl Gene ID of the maize waxy gene:

```
mysql -u anonymous -h mysql.ebi.ac.uk -P 4157
mysql> use zea_mays_core_18_71_6;
mysql> SELECT gene.stable_id
       FROM gene, xref
       WHERE gene.display_xref_id = xref.xref_id
         AND xref.display_label LIKE 'waxy';
```

MySQL pros and cons

- Quick and simple for basic queries
- Handy for verifying data
- Very good knowledge of underlying database schemas needed
- Schemas can change between releases
- Queries can quickly become very complex
  - Not recommended (and not supported) to retrieve sequences

Access method 3: Ensembl Perl API

- Mature, fully featured Perl API (Applications Programming Interface) for Ensembl resources
  - Perl: a commonly used programming language in bioinformatics, designed to make "easy thing easy and hard things possible"
  - Provides access to:
    - Genomic sequence
    - Genome features e.g. genes, translations
    - Annotation e.g. cross-references
  - [http://www.ensembl.org/info/docs/api/index.html](http://www.ensembl.org/info/docs/api/index.html)

Perl API

- Hierarchical object model
  - Retrieve a slice of a chromosome; then a gene on the slice; then a feature on the gene

Perl API example

Code snippet to retrieve the Ensembl Gene ID of the maize waxy gene:

```
my $gene_adaptor = $registry->
  get_adaptor( 'maize', 'core', 'gene' );

my $gene = $gene_adaptor->
  fetch_by_display_label( 'waxy' );

print $gene->stable_id, "\n";

object method argument
```
Perl API

- Used by the Ensembl analysis and annotation pipeline and the Ensembl web code
- Robust, reliable and well-supported

Needed:
- Perl
- BioPerl modules (version 1.2.3)
- Ensembl API modules
- Ability to code in Perl
- Knowledge of Perl APIs

Perl API pros and cons

- Most powerful way of accessing data; all data can be retrieved
- And even more, e.g. translations (which are not in the database)
- Stable through future releases
- Perl coding skills needed
- Knowledge of Perl APIs needed

Ensembl Genomes Perl API Extension

- Ensembl Genomes Perl API
  - Supplementary to Ensembl API
  - Provides additional lookup tools for finding genomes in Ensembl Bacteria (of which there are currently > 16,000):
    - Find genomes by name pattern
    - Find genomes by INSDC accession
    - Find genomes by taxonomy ID

Access method 4: REST API

- REpresentational State Transfer
  - is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements (Wikipedia)
  - A style for structuring URLs (i.e. web addresses) according to the content they contain
  - RESTful web service or RESTful web API
    - Allows users to access data simply by invoking the URL
    - Often returns a data structure defined in a simple grammar (e.g. JSON) which can be imported into an object in any programming language

Access method 4: REST API

- Endpoints provide data in different formats e.g.
  - JSON
  - XML
  - FASTA
  - GFF3

- Fully documented including examples in different languages
REST API example

URL to retrieve the Ensembl Gene ID of the maize waxy gene:

http://beta.rest.ensembl.org/xrefs/symbol/maize/waxy?content-type=text/xml

wget or curl can be used to retrieve data programatically from UNIX shell scripts

REST API pros and cons

- No specific knowledge necessary
- All languages can send RESTful requests
- Clean, intuitive API design
- Service still in its early stages
- Limited number of endpoints (but growing); not all data can be retrieved
- Access limitations; very heavy data requests will be bounced

Access method 5: BioMart

- A generic tool to facilitate the design and query of data warehouses
  - Data warehouses are databases designed to optimise the performance of certain commonly performed queries
  - May be less flexible than normalised schema
  - Less suitable for maintaining primary data (harder to automatically define constraints due to form of data model)
  - Nonetheless, can still be implemented within RDBMS
  - BioMart uses MySQL
- We have gene-centric and variant centric BioMarts for all Ensembl divisions
- BioMart comes with its own web interface

BioMart Web UI

But BioMart also comes with programmatic UIs

- Web services: at the touch of a button, a BioMart results page can generate a call to a RESTful or SOAPy web service that can be used to retrieve the same data programatically.
- Mart is also supported by a MySQL back-end, which can also be accessed directly (not recommended).
- MartShell – a tool that allows users to query the database from the command line using MQL – Mart Query Language – that works through the same abstraction layer as interactive queries
- Newer versions of Mart provide a SPARQL endpoint – not yet deployed by Ensembl Plants.

Access Method 6: Virtual Machines

- Download an environment containing all of Ensembl to run on your machine
- In effect, you are downloading/running a model of a computer
- As long as your computer can support running the VM, there should be no problem with library incompatibilities etc. - all the resources Ensembl needs are within the VM
- Increasingly, a model of choice for running web-based services (e.g. in cloud environments) – you don’t deploy into a platform, you deploy a whole platform
- We use OpenBox, an open source virtualisation platform
- http://ensemblgenomes.org/info/access/virtual_machine
Funding

- Ensembl Genomes Funded by
  - EMBL
  - EU (INFRAVEC, Microme, transPLANT, AllBio)
  - BBSRC (PhytoPath, wheat/barley/midge sequencing, UK-US collaboration, RNAcentral)
  - Wellcome Trust (PomBase)
  - NIH/NIAID (VectorBase)
  - NSF (Gramene collaboration)
  - Bill and Melinda Gates Foundation (wheat rust)

People

- James Allen, Irina Armean, Dan Bolser, Mikkel Christensen, Paul Davies, Lee Falin, Christoph Grabmüller, Kevin Howe, Malcolm Hinsley, Arnaud Kerhornou, Julia Khodanova, Eugene Kulesha, Nick Langridge, Dan Lawson, Mark McDowall, Uma Maheswari, Gareth Maslen, Michael Nuhn, Chuang Kee Ong, Michael Paulini, Helder Pedro, Anton Petrov, Dan Staines, Mary Ann Tuli, Brandon Walts, Gary Williams

- The vertebrate genomics team @ EBI (Paul Flicek)