

WormBase

Methods for Data Mining and Comparative Genomics

Todd W. Harris and Lincoln D. Stein

Summary

WormBase is a comprehensive repository for information on *Caenorhabditis elegans* and related nematodes. Although the primary web-based interface of WormBase (<http://www.wormbase.org/>) is familiar to most *C. elegans* researchers, WormBase also offers powerful data-mining features for addressing questions of comparative genomics, genome structure, and evolution. In this chapter, we focus on data mining at WormBase through the use of flexible web interfaces, custom queries, and scripts. The intended audience includes users wishing to query the database beyond the confines of the web interface or fetch data *en masse*. No knowledge of programming is necessary or assumed, although users with intermediate skills in the Perl scripting language will be able to utilize additional data-mining approaches.

Key Words: WormBase; *C. elegans*; AceDB; AcePerl; bioinformatics; data mining; comparative genomics.

1. Introduction

WormBase is a web-accessible database of *Caenorhabditis elegans* and related nematodes (1–4) that serves the *C. elegans* research community and makes advances in the field accessible to the broader biomedical community. Born from the pioneering work of the genome database system AceDB (5), WormBase has grown into an international consortium of researchers exploring methods of data analysis, warehousing, and visualization within the context of a highly developed model system.

The WormBase Consortium is composed of researchers at four locations: the California Institute of Technology (Caltech); the Genome Sequencing Center, Washington University (WashU); the Wellcome Trust Sanger Institute (Sanger); and Cold Spring Harbor Laboratory (CSHL). The duties of curating, analyzing,

From: *Methods in Molecular Biology*, vol. 351: *C. elegans: Methods and Applications*
Edited by: K. Strange © Humana Press Inc., Totowa, NJ

and presenting data are split between the groups. Caltech is primarily responsible for systematic curation of data from the published literature; WashU and Sanger are each responsible for curating half of the genome sequence (and associated gene models); CSHL is in charge of user-interface development and management of the website. Additionally, Sanger is responsible for assembling the release version of the database. This process merges the local databases and performs various analyses as necessary (e.g., BLAST analysis). The final database is distributed to CSHL where the data is made available to our users on the main website and official mirror sites.

WormBase contains a vast array of information mirroring the breadth and depth of the *C. elegans* research field. These data include the sequence and structure of the *C. elegans* and *Caenorhabditis briggsae* genomes (6,7); genetic information such as strains, alleles, and genetic maps; data from large-scale experiments such as genome-wide RNAi screens (8–11) and the ORFeome project (12,13); reagents such as antibodies, polymorphisms, and transgenes; gene and anatomy ontologies; curated functional descriptions; cellular data such as expression patterns and the complete lineage and connectivity maps (14–16); and comparative data such as orthologs and syntenic regions between species. Over the next year, WormBase will add the complete genomic sequence and annotations of three additional nematodes (*Caenorhabditis remanei*, *Caenorhabditis japonica*, and *Caenorhabditis* n. sp. PB2801).

In addition to web-based views of the data at <http://www.wormbase.org/>, a number of methods for data mining and comparative genomics are available for exploring the wealth of information housed at WormBase. These tools facilitate custom views of the data and the retrieval of specific datasets for further analyses.

1.1. Access Options

The most commonly used point of entry to WormBase is via the web interface at <http://www.wormbase.org/>. Additionally, WormBase has built a number of mirror sites to better serve our user base across the globe. A full listing of mirrors is available in the appendix; a list of current mirror sites is always displayed on the front page of WormBase. WormBase also maintains a development server (<http://dev.wormbase.org>) where new features and datasets are tested before being made available on the primary site. Users are welcome to use this site but are cautioned that the interface and data may not be stable. Finally, WormBase offers a publicly accessible data-mining server (aceserver.cshl.org; discussed in **Subheading 3.4.**).

1.2. Release Cycle, Versioning, and Data Freezes

WormBase follows a regular release cycle, currently one release approximately every 3 wk. This ensures timely access to new datasets and curator-con-

firmed annotations. Releases are versioned using the format “WSXXX” (e.g., WS100, WS101, and so on), with the current version of WormBase always displayed on the home page. Each new WormBase release is generated anew from the latest set of source databases from each WormBase group.

With every 10th release of the database, we create a data and software “freeze” and make it available in perpetuity at [http://\[ws_version\].wormbase.org/](http://[ws_version].wormbase.org/). These freezes serve two purposes. First, they ensure that the web interface for a particular freeze will continue to be available. This is important because the display layer of WormBase aggregates information in a way that is not readily available from the database itself. Furthermore, software development occurs in tandem with database releases and the two must be matched for consistent results. Second, freezes provide a stable referential release for ease of citation and verification. Collaborating groups can coordinate analyses by using the same version of the data. This is especially important when considering that the genome sequence undergoes small corrections periodically, potentially affecting any analysis dependent on the chromosomal location of features. Users conducting such studies are encouraged to use the most recent frozen version of WormBase.

1.3. Citing WormBase

How should users cite data obtained from WormBase? Direct links to the database can be cited as a full URL copied directly from a browser. Users should also make note of the particular release the data was obtained from (displayed on the home page). A sample citation might look like:

```
http://www.wormbase.org/db/gene/gene?name=unc-26;  
WormBase Release WS140.
```

Visit <http://www.wormbase.org/about/citing.html> for the most up-to-date information on citing WormBase.

1.4. Informatics Infrastructure

WormBase is comprised of three components critical to effective data mining: two database systems and the software that powers the website. AceDB is the primary database system. Curation centers maintain local datasets in AceDB and AceDB delivers much of the information displayed on the website. To facilitate fast queries of genomic annotations, some data is extracted from AceDB and deposited in secondary relational databases. These databases are created with the Bio::DB::GFF data schema and housed in MySQL. The web interface is generated by a collection of Perl scripts that interact with both the AceDB and MySQL databases.

1.4.1. The WormBase (AceDB) Data Model

Effective data mining requires an understanding of the underlying AceDB data model. AceDB uses an intuitive object-oriented structure that is well suited

for biological data. An AceDB data model consists of a series of structured classes stored in a human readable format. Each class can have a number of tags, which can point to various data types or cross-references to other objects. The presence or absence of some tags carry specific information. For example, the Gene class (**Fig. 1**) defines tags such as “Corresponding_CDS” and “Variation” to store references to the appropriate CDS and variation objects for a given gene. The “Live” tag has no subtags; its presence (or absence) indicates whether the gene is current (or has been retired). Individual tags can be accessed by name from instantiated objects, sparing programmers the tedium of complicated table joins typical of relational database systems.

The easiest way to become familiar with the data model used at WormBase is through the web interface. The contextual navigation bar on every data-bearing page on the website contains two links for exploring the data model. The “Schema” link displays the data schema for the class of the current object. The “Tree Display” link shows the schema populated with data from the current object. One can search for the model of a specific class from the home page by typing in a class name and selecting “Model” from the popup menu. Questions about the model should be addressed to help@wormbase.org. Additional information on the structure of AceDB data models is available at <http://www.acedb.org/>.

1.4.2. Overview of Prominent Classes

The AceDB database at WormBase spans almost 100 classes (prominent classes are described in **Table 1**). Although class names usually reflect the type of objects they are meant to represent, this is not always transparent because of generalizations in the data model or historical constraints. For example, the Gene class includes a variety of biologically distinct concepts, such as genetic loci, noncoding RNAs, and predicted genes. An easy way to become familiar with the various classes at WormBase is through the Class Browser page at http://www.wormbase.org/db/searches/class_query.

1.4.3. Object Identifiers

All objects in WormBase drawn from the AceDB database have an associated *class* and *name*. To discover the class of a particular object, search for it from the main page using the “Anything” option. The class and name will be displayed in the URL. For most classes, object names correlate with the most common name of the object itself. Recently, however, WormBase has moved towards serialized identifiers for a number of classes. This makes data management and versioning substantially easier and allows for distinct objects to have the same public name. These classes typically have a *_name companion class that serves to map publicly used names to the serialized object name. Classes using serialized identifiers include Gene, RNAi, Paper, and Variation.

```

?Gene Evidence #Evidence
SMap S_parent UNIQUE Sequence UNIQUE ?Sequence XREF Gene_child
Identity Version UNIQUE Int
    Name CGC_name UNIQUE ?Gene_name XREF CGC_name_for #Evidence
    Sequence_name UNIQUE ?Gene_name XREF Sequence_name_for
    Molecular_name ?Gene_name XREF Molecular_name_for
    Other_name ?Gene_name XREF Other_name_for #Evidence
    Public_name UNIQUE ?Gene_name XREF Public_name_for
    Species UNIQUE ?Species
    Live
Gene_info Gene_class UNIQUE ?Gene_class XREF Genes
    Laboratory ?Laboratory
    Reference_allele ?Variation Text
    Allele ?Variation XREF Gene #Evidence
    Phenotype ?Text #Evidence
    Strain ?Strain XREF Gene
    GO_term ?GO_term XREF Gene ?GO_code #Evidence
    Contained_in_operon ?Operon XREF Contains_gene
    Ortholog ?Gene XREF Ortholog ?Species #Evidence
    Structured_description Provisional_description ?Text
    Detailed_description ?Text #Evidence
    Concise_description ?Text #Evidence
    Other_description ?Text #Evidence
    Biological_process ?Text #Evidence
    Molecular_function ?Text #Evidence
    Expression ?Text #Evidence
Molecular_info Corresponding_CDS ?CDS XREF Gene #Evidence
Experimental_info RNAi_result ?RNAi XREF Gene
    Expr_pattern ?Expr_pattern XREF Gene
    Drives_Transgene ?Transgene XREF Driven_by_gene
    Transgene_product ?Transgene XREF Gene
    Rescued_by_transgene ?Transgene
    Gene_regulation Trans_regulator ?Gene_regulation
    Antibody ?Antibody XREF Gene
    Microarray_results ?Microarray_results XREF Gene
Map_info Map UNIQUE ?Map XREF Gene #Map_position
    Hide_under ?Gene XREF Representative_for
    Representative_for ?Gene XREF Hide_under
    Negative_clone ?Clone XREF Negative_gene #Evidence
    Mapping_data 2_point ?2_point_data
    Multi_point ?Multi_pt_data
    Pos_neg_data ?Pos_neg_data
Reference ?Paper XREF Gene
Remark ?Text #Evidence
Method UNIQUE ?Method

```

Fig. 1. A partial listing of the AceDB Gene class. Objects referenced in the model are preceded by a “?”. Tags may store multiple items to their right, including text and object cross-references. Cross-references (denoted by “XREF”) serve to associate specific tags in related objects to the current object. Note that some tags accept no data to their right (e.g., “Live”). These tags carry information according to their presence or absence. Another important element used throughout the data model is the evidence hash (denoted using the nomenclature “#Evidence”). Evidence hashes store information pertaining to the source of individual pieces of data.

Table 1
Prominent Classes in the AceDB Data Model at WormBase

Object	Name	Class	Description/notes
A predicted gene	A cosmid dot name, such as JC8.10a	Gene_name	The Gene_name class serves to relate commonly used gene names to the serialized names used for Gene objects.
A genetic locus	A CGC standard locus name, such as lin-29	Gene_name	Like predicted genes, the Gene_name class relates publicly used names to the Gene class
A gene	A serialized WBGeneID	Gene	Gene objects are the top-level containers for genetic loci, molecularly identified genes, and RNAs.
A Genbank Accession Number (protein or nucleotide)	The accession number	Accession_Number	This will retrieve an Accession_Number object which is a list of WormBase names that correspond to that accession number.
A protein	A wormpep accession number, preceded by "WP.": as in WP:CE28571	Protein	The corresponding Gene object of a protein can be fetched by searching the Gene_name class.
A clone	The cosmid or YAC name, for example T20H4	Clone	
A cell	A cell name, such as M1	Cell	
An RNAi experiment	A WBRNAi identifier	RNAi	Like the Gene class, WormBase uses serialized identifiers for the RNAi class. Previously used experiment identifiers (such as JA:59E12.2) are retained under the History_name tag.
An allele or polymorphism	An allele name, such as e345 or a WBVariation identifier	Variation	The Variation class uses serialized identifiers.
Genomic sequence	The cosmid or YAC from which the sequence was obtained, for example Y67H2	Genomic_Sequence	A generic class name of sequence is also recognized

2. Materials

Data mining at WormBase requires only modest computational power. Any current personal computer will be suitable for web-based data-mining approaches. Users wishing to access the resource programmatically using the methods described next will need access to an operating system that supports the Perl programming language.

3. Methods

WormBase offers four data-mining options to meet the needs of our users. In increasing order of complexity and flexibility are web-based data mining pages, query languages, programmatic mining of the web interface, and scriptable access via the AcePerl and Bio::DB::GFF programming interfaces. In the listings that follow, query and code examples are displayed in a **fixed width font**. The WormBase data-mining archive (http://www.wormbase.org/data_mining/) contains additional sample queries and scripts as well as expanded versions of the scripts described beginning in **Subheading 3.2.1**.

3.1. Web-Based Data-Mining Options

WormBase offers multiple web-based data-mining options. WormMart (the WormBase implementation of BioMart (17); <http://www.wormbase.org/BioMart/martview>) is the newest addition to our data-mining repertoire. WormMart provides a flexible, user-friendly interface for retrieving select data from WormBase *en masse*. Users begin by selecting a reference release of the database followed by a point of focus (e.g., a gene). Results can be restricted to a list of identifiers or chromosomal or genetic map positions. Next, a series of filters can be enabled further restricting the results returned. For example, a user might choose to include only those genes that have alleles. Finally, a broad range of data can be selected that should be included in the output.

The “Batch” scripts (Batch genes and Batch sequences) offer access to annotations and sequences, respectively. Both pages support searching with a list of gene names or identifiers, the use of wildcards, and HTML or plain text output. It is anticipated that WormMart will eventually replace the “Batch” scripts. WormBase also provides a number of tools for querying specific datasets, such as RNAi results, expression patterns, and microarray data. These tools are described in more detail on the WormBase site map (http://www.wormbase.org/db/misc/site_map).

3.2. Query Languages

WormBase offers two query languages, both of which were developed as part of the underlying AceDB database system. Queries written in these languages can be submitted to the database through web forms. Because query

languages work on the database directly, users can retrieve data that may not be displayed on the website, in a form amenable to further processing. Successful queries require familiarity with the underlying data model.

3.2.1. WormBase Query Language

Of the two query languages offered at WormBase, the WormBase query language (WQL) is the easier to use. In general, WQL queries take the structure of “find CLASS [PROPERTIES]” where CLASS is the class of interest and PROPERTIES are filters to apply to the returned keyset. WQL queries support the use of logical operators (NOT, OR, AND, and so on) and wildcard characters (? and *). The following examples can be submitted at http://www.wormbase.org/db/searches/wb_query. This page can also be found from either the “Site Map” or “Searches” link on the primary navigation bar at the top of every page.

Fetch all *C. elegans* genes in the database.

```
find Gene WHERE Species="Caenorhabditis elegans"
```

Find a list of genes that might encode actin proteins.

```
find Gene WHERE Public_name="act-*
```

Fetch all genes that are cloned and have alleles. Here we test for the existence of the “Allele” and “Corresponding_CDS” tags in Gene objects. This will return genetically defined and cloned genes, as well as those identified through reverse genetics.

```
find Gene WHERE Allele AND Corresponding_CDS
```

To connect individual queries together, use a semicolon. The second query will be run against the results of the first query. The following example will find all CDSs that are confirmed by the presence of a cDNA.

```
find CDS WHERE Prediction_status="Confirmed"; Matching_cDNA
```

To traverse from one class, to another, use the “Follow” command. The following command will fetch all genes that have an Emb RNAi phenotype on Chromosome III. In this query, we first find all RNAi experiments that result in an Emb phenotype, then “Follow” these results to a corresponding set of Gene objects. Finally, we select a subset of the initial results based on chromosomal location.

```
find RNAi WHERE Phenotype="Emb"; Follow Gene; Map="III"
```

When querying subtags, the primary tag must be specified and the subtag preceded by a “#.” The “HERE” clause lets an additional condition be applied to the same tag. It is important to note that the query operators AND, OR, NOT, and HERE must be in all capitals; the remainder of the query can be in lowercase. This query will find all Variation objects between two genetic map points, and then selects a subset of polymorphisms based on the presence of the “SNP” tag.

```
find Variation Map="I" # (Position >= 5.0 AND HERE <= 10.0) AND (SNP
```

3.2.2. AceDB Query Language

The AceDB query language uses an expanded syntax that facilitates more complicated queries. In particular, AceDB query language offers a structured query language-style syntax that lets users select the columns (in this case tags) to include in the output from a given key set. The following example queries against the database can be submitted at http://www.wormbase.org/db/searches/aql_query.

Fetch all genes in the database. Here we use an alias (a) to represent instances of Gene objects fetched by the query.

```
select a from a in class Gene
```

Find all genes in the database, including their WormBase WBGeneID and their more commonly used public name.

```
select a,a->Public_name from a in class Gene
```

Find all alleles that have phenotype data but lack sequence information. This example checks for the presence and absence of the “Phenotype” and “Sequence” tags, respectively, returning a list of variations.

```
select all class Variation where exists_tag->Phenotype and
not exists_tag->Sequence
```

Fetch all polymorphisms that occur within genes, reporting the polymorphism name and containing gene. This query checks for the presence of the “SNP” and “Gene” tags of Variation objects in a boolean context.

```
select a,a->Gene from a in class Variation where
exists a->SNP[0] and exists a->Gene[0]
```

3.3. Programmatically Mining the Web Interface

A first step toward programmatic data mining at WormBase is to create scripts that extract data directly from pages on the website. This approach is often referred to as screen-scraping. Screen-scraping scripts can be created with limited knowledge of a programming language such as Perl. Although not an official WormBase data mining method, the ease with which these scripts can be created make them a viable strategy for fetching data from the site.

Screen-scraping offers several advantages over more direct data-mining approaches. Users need not be familiar with the data model in order to write effective scripts. Furthermore, the web displays at WormBase often integrate data from several different classes in a single comprehensive display. Often it is simpler to screen-scrape this data rather than trying to recreate the logic of a script using programmatic interfaces to the database. Because screen-scraping relies on parsing the HTML of a page, scripts are susceptible to changes in the document. Also, because they are limited to data presented on the visual interface, they are not as flexible as approaches that mine the data-

base directly. Nevertheless, screen-scraping can be a quick way to access a collection of data presented on the website.

The following script takes a list of loci (hard-coded in the script itself) and fetches the “Concise Descriptions” displayed on the “Gene” page (<http://www.wormbase.org/db/gene/gene>). See **Notes 1** and **2** for information on running this script on your system.

```

1  #!/usr/bin/perl -w
2  # This script demonstrates one way to "screen scrape" WormBase
3
4  use strict;
5  use WWW::Mechanize;
6
7  # The URL of the target page
8  use constant URL =>
9      'http://aceserver.cshl.org/db/gene/gene?name=';
10
11 # A list of genes to fetch
12 my @genes = qw/unc-2 unc-26 unc-70 unc-119 dyn-1 vab-3/;
13
14 foreach my $gene (@genes) {
15     my $agent = WWW::Mechanize->new();
16     $agent->get(URL . $gene); # Create the full URL
17     $agent->success
18         or die "Sorry, I couldn't fetch the page for $gene: $!";
19     my $content = $agent->content;
20
21     # Parse out the Concise Description.
22     my ($description) =
23         ($content =~ /Concise\sDescription.*?body">(.*?)\s\[//);
24
25     print "$gene\t$description\n";
26     sleep 3;
27 }

```

This script uses standard Perl notation (*see Note 3*), executing the following steps. Line 5 makes functions provided by the `WWW::Mechanize` library available to the script. In lines 8–9, a constant is defined containing a fragment URL of the page to mine for information. In line 12, we specify a list of genes to fetch concise description for. Line 14 begins a loop iterating over each of the values in the `@genes` array. Line 15 creates a `WWW::Mechanize` object which will handle interactions with the website. We store the object in the `$agent` variable. In line 16, we use the `WWW::Mechanize` agent to fetch the page for the current gene by concatenating the URL constant value to the current value of `$gene`. Lines 17–18 verify that the request was successful using the `success()` method of `WWW::Mechanize` in boolean context. If the request was successful, line 19 fetches the content of the page, storing it in the `$content` variable. Note that `$content` contains HTML formatting, not just the text visible on the web page. Lines 22 and 23 are the most critical elements of this script. This regular expression parses out the text that comes immediately after “Concise Description” on

the web page, storing it in the `$description` variable. Line 25 prints out the gene name and description parsed from the page. Lines 26 and 27 are the end of the loop, with a 3-s delay before the next iteration.

Out of courtesy to other users, we request that users screen-scraping WormBase direct their scripts to <http://aceserver.cshl.org/> and that they include a 3-s delay between requests. Our full acceptable use policy is available at http://www.wormbase.org/about/acceptable_use.html.

3.4. Programmatic Access to AceDB Using AcePerl

Although screen-scraping approaches are fast and easy to implement, they are limited to data that is visible on the web interface. The AcePerl Perl module (**19**) circumvents these limitations by providing direct access to the AceDB database at WormBase. AcePerl is particularly useful for data-mining tasks seeking text-based annotations or those that need to cross several different database classes. Using AcePerl requires beginner to intermediate knowledge of Perl including a familiarity with the Perl object-oriented idiom.

WormBase offers a publicly accessible data-mining server open to scripts using AcePerl. To use this server, scripts should be directed to aceserver.cshl.org, port 2005. Alternatively, queries can be directed against a local data source. The following script demonstrates how to fetch all known alleles for all *C. elegans* genes from *aceserver* (see **Note 4** for information on the WormBase representation of genes). In order to develop AcePerl scripts or test the example script, the AcePerl module must be installed (see **Note 5**).

```

1  #!/usr/bin/perl -w
2  use strict;
3  use Ace;
4
5  my $DB = Ace->connect(-host => 'aceserver.cshl.org',
6                      -port => '2005')
7                      or die "Couldn't connect to aceserver: $!";
8
9  my @genes = $DB->fetch(-query=>
10     qq{find Gene where Species="Caenorhabditis elegans"});
11
12 foreach my $gene (@genes) {
13     my $name = $gene->Public_name || $gene;
14     my @alleles = $gene->Allele;
15     print join("\t", $name, @alleles), "\n";
16 }

```

This script uses standard Perl notation (see **Note 3**), executing the following steps. Line 3 makes functions provided by the AcePerl library available to the script. Lines 5–7 establish a connection to the appropriate database, exiting the script if a connection cannot be established. In lines 9–10, a query is executed on the AceDB database, fetching all *C. elegans* genes. See **Note 6** for an example of fetching genes by their public name. Line 12 begins a loop iterating over

all genes returned by the query. In line 13, we set the name of the gene to the “Public_name” tag of the object (if it exists) or else it becomes the name of the object itself. Line 14 fetches all known alleles for the gene and stores them in the `@alleles` array. Line 15 prints the name of the gene and a list of its alleles to standard output and the loop is ended.

3.5. Programmatic Access to GFF Databases Using `Bio::DB::GFF`

To retrieve genomic annotations (including anything that may have sequence coordinates), use the `Bio::DB::GFF` Perl API (part of the BioPerl project [19]). The `Bio::DB::GFF` module provides methods for creating and querying relational databases of genomic annotations. These databases are created from tab-delimited files of genomic annotations in the GFF format. Files contain one annotation per line with fields describing its reference sequence (typically a chromosome), its source (how the annotation was derived), its method (the type of annotation), and various other attributes such as strand, phase, and start and stop coordinates. WormBase currently uses two `Bio::DB::GFF` databases, one each for *C. elegans* and *C. briggsae* genomic annotations. The BioPerl suite of modules must be installed (available from CPAN or www.bioperl.org) in order to develop scripts utilizing `Bio::DB::GFF` databases.

Features may be fetched from `Bio::DB::GFF` databases using feature names, coordinates, ranges, or the source and method of the annotation, usually presented as a method:source couplet. For example, to fetch the coordinates of all CDSs in the database, one might query for “curated:CDS.” The GFF methods and sources currently in use at WormBase are displayed at http://www.wormbase.org/db/misc/database_stats. Grouped features (like the introns and exons of a transcript; see Note 7) can be aggregated together during retrieval. Features can be reported in relative or absolute coordinates (and converted between the two as desired).

The following example script can generically fetch any sequence feature from the *C. elegans* GFF database on the WormBase data-mining server (aceserver.cshl.org), optionally fetching the DNA of the feature in FASTA format (see Notes 8 and 9 for information on how to run this script).

```

1 #!/usr/bin/perl -w
2
3 use strict;
4 use Bio::DB::GFF;
5
6 my $feature = shift;
7 my $dna = shift;
8 $feature || die "You must specify a feature to fetch...\n";
9
10 # Establish a connection to the appropriate data source
11 my $db = Bio::DB::GFF->new(-dsn =>
12     'dbi:mysql:elegans:aceserver.cshl.org',
```

```

13                                     -user => 'anonymous')
14     || die "Couldn't establish a connection to DSN: $!";
15
16 # Fetch an iterator of the requested feature
17 my $iterator = $db->get_seq_stream(-type => $feature);
18 while (my $feature = $iterator->next_seq) {
19
20     # Create an informative header
21     my $name = $feature->name;
22     my $type = $feature->type;
23     my ($start,$stop) = ($feature->start,$feature->stop);
24     my $refseq = $feature->sourceseq;
25     my $header = "$name ($type; $refseq:$start..$stop)";
26
27 # If requested, fetch sequence of the feature in FASTA
28 if ($dna) {
29     my $seq = to_fasta($feature->dna);
30     print ">$header\n", $seq, "\n";
31 } else {
32     print "$header\n";
33 }
34 }
35
36 # This subroutine converts a dna string into fasta format
37 sub to_fasta {
38     my $sequence = shift;
39     return if ($sequence =~ /^>(.)$/m); # Return if already in FASTA
40     $sequence =~ s/(.{80})/$1\n/g; # Carriage return
41     return $sequence;
42 }

```

This script uses standard Perl notation (*see Note 3*), executing the following steps. Line 4 makes the Bio::DB::GFF library available to the script. Lines 6–8 accept input from the command line: the required feature to fetch, and an optional flag to fetch DNA if desired. In lines 11–14, the script attempts to connect to the GFF database on `aceserver.cshl.org` using the `connect()` method of Bio::DB::GFF, exiting if a connection cannot be established. Line 16 fetches a feature stream from the database of the desired feature. In line 18, a loop is begun that iterates over all features of the specified type. In lines 21–24, a variety of information about the current feature is fetched and stored in variables; these are used to create a FASTA style header in line 25. Lines 28–34 generate the primary output of the script. If the “dna” option is specified on the command line, the sequence of the feature is fetched in FASTA format and printed along with the header in lines 29 and 30; if not, the header line itself is printed in line 32. The primary loop ends in line 34. Lines 37–42 comprise a subroutine that converts a sequence into FASTA format.

3.6. Running WormBase Locally

For users requiring high speed—such as those conducting large amounts of data-mining experiments—a local installation of WormBase is an attractive

option. Local installations are not limited by server or network congestion or even the need for Internet access. An installation can be limited to the AceDB and GFF databases in use at WormBase, or can include the software that drives the WormBase site as well. Local installations of WormBase also benefit from the ability to use AceDB directly (including the graphical and text-based interfaces to AceDB databases, *xace* and *tace*).

Hardware requirements differ based on the components of WormBase you choose to install and how the site will be used. Minimally, a 1-Ghz G4 or 900 Mhz Pentium III processor are required for acceptable performance, running either Mac OS X or a Unix/Linux variant. WormBase can run acceptably for a small number of users with 1 GB of memory, but a minimum of 4 GB memory is recommended for server installations. The AceDB and GFF databases currently require about 10 and 4 GB of disk space, respectively. Users should expect disk space requirements to grow steadily with the scheduled addition of three new genomes. These relatively modest requirements mean that a desktop machine or laptop can be suitable for running a local copy of WormBase. For installations serving multiple users, a server class machine should be considered. We refer users to the current documentation describing the installation procedure at <http://www.wormbase.org/docs/INSTALL.html>.

For convenience, we provide prepackaged files of the database components of WormBase for every release. Currently, this consists of four files: the AceDB database, the *C. elegans* GFF database, the *C. briggsae* GFF database, and databases to support the BLAST and BLAT searches. These packages greatly simplify local installations. The current versions of these files, as well as documentation on how to install them can be found at ftp://ftp.wormbase.org/pub/wormbase/mirror/database_tarballs/.

The software that drives WormBase can be obtained through several avenues. The recommended method is to retrieve the software by anonymous rsync access. This ensures direct synchronization to the current WormBase site. Alternatively, compressed archives of the software corresponding to each data release are available on the WormBase FTP site. Finally, we offer anonymous CVS access to the WormBase code (*see* <http://www.wormbase.org/docs/INSTALL.html> for additional information). The Bio::GMOD Perl module (available on CPAN) contains a number of scripts for automating updates of a WormBase installation.

3.7. Linking to WormBase

To generate a link to an object in WormBase, specify the class and name of the object using a URL with the following structure:

```
http://www.wormbase.org/db/get?name=WBGene00006763;class=gene
```

You can also fetch an XML representation of an object by linking to the following URL:

```
http://www.wormbase.org/db/misc/xml?name=WBGene00006763;class=Gene
```

And similarly for a text-only display of the object:

```
http://www.wormbase.org/db/misc/text?name=WBGene00006763;class=Gene
```

To link to an image of the genome, use a URL of the following format:

```
http://www.wormbase.org/db/seq/gbrowse/wormbase?name=unc-26
```

The “name” parameter can accept almost any item that might have coordinates on the physical map, or a physical span (e.g., III:30000..50000). See http://www.wormbase.org/data_mining/linking.html for the most current linking rules.

3.8. Displaying Remote Annotations on the Genome Browser

An often overlooked but very useful feature of the Genome Browser is the ability to display remotely hosted annotations within the context of annotations housed at WormBase (20). Annotations can be entered as plain text through a web form or by uploading a file as described in the “Upload your own annotations” link at the bottom of the Genome Browser. Uploaded annotations are also included in images generated from the “Publication Quality Image” link. This option creates a scalable vector graphics image of the current genome span. Scalable vector graphics images can be edited in vector-based graphics applications, enlarged or compressed with no loss in quality, and submitted as high quality figures for publication.

To publish preliminary data sets or remote annotations, users may make use of the distributed annotation system (DAS) (21). Using DAS, annotations are hosted on a remote server but displayed in the context of WormBase by a DAS client (in this case the Genome Browser). These annotations can be made public simply by notifying WormBase of the desire to make the data available. We can assist in creating a custom option that allows WormBase users to enable display of the remote data.

3.9. Tools for Comparative Genomics

WormBase provides several useful tools and precomputed datasets for comparative genomics analyses. First, WormBase houses the entire *C. elegans* and *C. briggsae* genomic sequences and predicted gene and protein sets. Second, pre-calculated reciprocal best mutual match BLASTP orthologs between *C. briggsae* and *C. elegans* are displayed on the Gene Summary page (when known). Finally, WormBase displays nucleotide level alignments between *C. elegans* and *C. briggsae* on both the Genome Browser and the Synteny Browser. We will extend the utility of WormBase for comparative genomics analyses in the coming year with the addition of three additional *Caenorhabditis* genomes.

Nucleotide level alignments identified through use of the WABA algorithm (22) are displayed on both the Genome Browser and the Synteny Browser. WABA distinguishes between “coding,” “strong,” and “weak” alignments. Prior to display, these alignments are post-processed to generate a series of syntenic blocks. On the Genome Browser, such merged syntenic blocks (7) are displayed under the “*Briggsae* alignments” track using the following color scheme to distinguish the strength of the alignment: dark blue (coding), light blue (strong), and gray (weak). Because merged blocks can contain gaps where there was no distinct nucleotide alignment, a single syntenic segment on the Genome Browser may not be continuous. Such gaps within a given syntenic block are represented by a dashed gray line. Clicking on one of these alignments will take the user to the Synteny Browser where syntenic blocks can be viewed in relation to both genomes simultaneously.

4. Notes

1. To execute the Perl example scripts, a system with Perl v5.6.0 or greater is required. Scripts can be entered manually into a text editor or word processing program and saved to disk, or downloaded directly from http://www.wormbase.org/data_mining/ in order to avoid data entry errors. If typing in the example scripts manually, do not include the listed line numbers. After the script has been entered or downloaded it needs to be made executable by entering the following commands at a command line prompt.

```
shell> chmod 770 example_script.pl
shell> ./example_script.pl
```

To capture the output of any script, redirect standard output to a filename using the following notation.

```
shell> ./example_script.pl > output.txt
```

2. This example script uses the WWW::Mechanize module, available from CPAN. This can be installed from a command line prompt (denoted by “`shell>`” below), using the following commands.

```
shell> sudo perl -MCPAN -e shell
cpan> install WWW::Mechanize
cpan> quit
```

You will need superuser privileges on your system in order to install WWW::Mechanize as indicated. Consult your system administrator if you do not have these privileges. See **Note 1** for how to execute the script.

3. The Perl programming language uses the following general conventions. Perl scripts typically begin with the notation “`#!/usr/bin/perl`” which indicates that the file is a Perl script and where the Perl binary is installed on your system. Lines in which the first character is a “`#`” are comments in the code and ignored by the Perl interpreter (with the exception of the first line). The “`use strict`” pragma, although not required, specifies that all variables must be explicitly declared. Lines in a Perl script can continue onto subsequent lines; a semi-colon indicates the end of a line. Perl uses three variable types common to most pro-

gramming languages: scalars, arrays, and associative arrays. Scalar variable names are preceded with a “\$” (e.g., `$variable`) and can contain either strings or numeric values. Arrays begin with an “@” (e.g., `@list`). Associative arrays or hashes contain key-value pairs and begin with a “%” (e.g., `%hash`). For additional information, consult a tutorial on Perl (23) or the built in Perl documentation on your system.

4. When implementing a data mining approach, it is important to consider how WormBase represents genes, CDSs, and transcripts in the data model. WormBase has implemented a unified Gene class intended as a top-level container for all data logically associated to a particular unit of a chromosome. Each Gene object may have zero, one or many associated CDSs listed under the “Corresponding_CDS” tag, corresponding to alternative splices in the coding sequence. Genes lacking CDSs are usually uncloned genes, noncoding genes, or pseudogenes. Finally, each CDS object may have one or more associated transcript objects (stored under the “Corresponding_transcript”) corresponding to alternative splicing in UTRs.
5. In order to execute this script, you will need to install AcePerl. This is done most easily via the Perl CPAN module. From a command line prompt, the following stanzas will install AcePerl on most 32-bit Unix-based systems.

```
shell> sudo perl -MCPAN -e shell
cpan> install Ace
// choose options 2 or 3, accept all other defaults
```

Once installed, you can read more about using AcePerl using the following command.

```
shell> perldoc Ace
```

6. To fetch Gene objects from AceDB using commonly used locus or molecular identifiers, first query the “Gene_name” class, then fetch the corresponding Gene object:

```
$gene_name = $DB->fetch(Gene_name => 'unc-70');
$gene = $gene_name->Public_name_for;
```

The “Gene_name” class provides a convenient mechanism for mapping the many publicly used names for a gene to a single gene object. Using the example above as a guide, one can search for gene objects using protein or locus names or molecular identifiers.

7. An understanding of how sequence features are represented in the GFF data schema is essential for data mining. In the current GFF databases, WormBase represents a single span corresponding to the maximal extents of a gene as Corresponding_transcript:Transcript (source:method). Similarly, a single full length span corresponding to just the CDS is listed as curated:CDS. The component parts of a full length transcript are Coding_transcript:intron, Coding_transcript:coding_exon, Coding_transcript:three_prime_UTR and Coding_transcript:five_prime_UTR. Note that this implementation is subject to change with the introduction of GFF3, which provides a more robust mechanism for grouping features than is possible with the GFF2 specification.

8. The `Bio::DB::GFF` example script accepts two positional arguments. The first is the `method:source` of the feature to fetch from the database. The optional second argument acts as a boolean flag to fetch the DNA of the feature in FASTA format. The following command will fetch all exons from *C. elegans*, printing their absolute start and stop positions and their sequence.

```
shell> gff_example.pl coding_exon:Coding_transcript dna
```

9. To utilize the public data-mining server for `Bio::DB::GFF` scripts, users should set the “`dsn`” option to “``dbi:mysql:aceserver.cshl.org``,” and the username to “``anonymous``.” The password field is not required. The example script sets these values automatically.

```
my $db = Bio::DB::GFF->new(-dsn => 'dbi:mysql:aceserver.cshl.org',
-user => 'anonymous');
```

Appendix: Select URLs

The primary WormBase server	http://www.wormbase.org/
Mirror sites:	
Caltech (Pasadena, CA)	http://caltech.wormbase.org/
IMBB (Crete, Greece)	http://imbb.wormbase.org/
Development server	http://dev.wormbase.org/
Data mining server	aceserver.cshl.org
	Ports:
	Web: http://aceserver.cshl.org/
	Bio::DB::GFF scripts: 3306
	AcePerl scripts: 2005
FTP site	ftp://ftp.wormbase.org/
Acceptable use policy	http://www.wormbase.org/about/acceptable_use.html
Citing WormBase	http://www.wormbase.org/about/citing.html
Data mining archive	http://www.wormbase.org/data_mining/
Installation guide	http://www.wormbase.org/docs/INSTALL.html
Data submission, questions, comments	help@wormbase.org

Acknowledgments

The authors wish to thank Keith Bradnam, Payan Canaran, Jack Chen, and Igor Antosheckin for critical readings of the manuscript. WormBase is supported by grant P41-HG02223 from the US National Human Genome Research Institute and the British Medical Research Council.

References

- Stein, L., Sternberg, P., Durbin, R., Thierry-Mieg, J., and Spieth, J. (2001) WormBase: network access to the genome and biology of *Caenorhabditis elegans*. *Nucleic Acids Res.* **29**, 82–86.

2. Harris, T. W., Lee, R., Schwarz, E., et al. (2003) WormBase: a cross-species database for comparative genomics. *Nucleic Acids Res.* **31**, 133–137.
3. Harris, T. W., Chen, N., Cunningham, F., et al. (2004) WormBase: a multi-species resource for nematode biology and genomics. *Nucleic Acids Res.* **32**, 411–417.
4. Chen, N., Harris, T. W., Antoshechkin, I., et al. (2005) WormBase: a comprehensive data resource for *Caenorhabditis* biology and genomics. *Nucleic Acids Res.* **33**, 383–389.
5. Durbin, R. and Thierry-Mieg, J. (1991) A *C. elegans* database. Documentation and code available from www.acedb.org. Accessed: April 7, 2006.
6. The *C. elegans* Genome Sequence Consortium. (1998) Genome sequence of the nematode *C. elegans*: a platform for investigating biology. *Science* **282**, 2012–2018.
7. Stein, L. D., Bao, Z., Blasiar, D., et al. (2003) The genome sequence of *Caenorhabditis briggsae*: a platform for comparative genomics. *PLoS Bio.* **1**, E45.
8. Fraser, A. G., Kamath, R. S., Zipperlen, P., Martinez-Campos, M., Sohrmann, M., and Ahringer, J. (2000) Functional genomic analysis of *C. elegans* chromosome I by systematic RNA interference. *Nature* **408**, 325–330.
9. Gonczy, P., Echeverri, C., Oegema, K., et al. (2000) Functional genomic analysis of cell division in *C. elegans* using RNAi of genes on chromosome III. *Nature* **408**, 331–336.
10. Kamath, R. S., Fraser, A. G., Dong, Y., et al. (2003) Systematic functional analysis of the *Caenorhabditis elegans* genome using RNAi. *Nature* **421**, 231–237.
11. Sonnichsen, B., Koski, L. B., Walsh, A., et al. (2005) Full-genome RNAi profiling of early embryogenesis in *Caenorhabditis elegans*. *Nature* **434**, 462–469.
12. Reboul, J., Vaglio, P., Rual, J. F., et al. (2003) *C. elegans* ORFeome version 1.1: experimental verification of the genome annotation and resource for proteome-scale protein expression. *Nat. Genet.* **34**, 35–41.
13. Lamesch, P., Milstein, S., Hao, T., et al. (2004) *C. elegans* ORFeome version 3.1: increasing the coverage of ORFeome resources with improved gene predictions. *Genome Res.* **14**, 2064–2069.
14. Sulston, J. E. and Horvitz, H. R. (1977) Post-embryonic cell lineages of the nematode, *Caenorhabditis elegans*. *Dev. Biol.* **56**, 110–156.
15. Sulston, J. E., Schierenberg, E., White, J. G., and Thomson, J. N. (1983) The embryonic cell lineage of the nematode *Caenorhabditis elegans*. *Dev. Biol.* **100**, 64–119.
16. White, J. G., Southgate, E., Thomson, J. N., and Brenner, S. (1986). The structure of the nervous system of *Caenorhabditis elegans*. *Philos. Trans. R. Soc. Lond. (B. Biol. Sci.)*. **314**, 1–340.
17. Additional information on BioMart is available at <http://www.ebi.ac.uk/biomart/>. Accessed: April 7, 2006.
18. Stein L. D. and Thierry-Mieg, J. (1998) Scriptable access to the *Caenorhabditis elegans* genome sequence and other ACEDB databases. *Genome Res.* **8**, 1308–1315.
19. Stajich, J. E., Block, D., Boulez, K., et al. (2002) The BioPerl toolkit: Perl modules for the life sciences. *Genome Res.* **12**, 1611–1618.

20. Stein, L. D., Mungall, C. Shu, S., et al. (2002) The generic genome browser: a building block for a model organism system database. *Genome Res.* **12**, 1599–1610.
21. Dowell, R. D., Jokerst, R. M., Day, A., Eddy, S. R., and Stein L. (2001) The distributed annotation system. *BMC Bioinformatics* **2**, 7 Epub, Oct 10.
22. Kent, W. J., and Zahler, A. M. (2000) Conservation, regulation, synteny, and introns in a large-scale *C. briggsae*–*C. elegans* genomic alignment. *Genome Res.* **8**, 1115–1125.
23. Schwarz, R.L. and Christiansen, T. (2001). *Learning Perl, 2nd Ed.*, O'Reilly, Sebastapol, CA.