GeeFu v0.1.0
An extensible RESTful Ruby on Rails web-service database for GFF data.

Dan MacLean
dan.maclean@tsl.ac.uk

Contents

1 About GeeFu

GeeFu is a Ruby on Rails [1] based RESTful web-service application that serves genome feature data on request. It is ideally suited to serving large amounts of data such as those from high-throughput sequencing experiments. It can use bam files from next-generation alignments directly as a data source. Simple web-requests via URLs can be used to return data in common web-formats like XML or JSON. GeeFu can be used in conjunction with web-service viewports such as AnnoJ [2] to create very fast, data-rich, attractive, RDBMS agnostic genome browsers and can be easily extended into custom web-applications using the powerful Rails framework.

1.1 Web services and GeeFu

Wikipedia [3] describes web services as follows

> In common usage the term refers to clients and servers that communicate over the Hypertext Transfer Protocol (HTTP) protocol used on the web. Such services tend to fall into one of two camps: Big Web Services and RESTful Web Services.

The use of web services means that data sources and application components can be distributed across the web. By maintaining standard data exchange formats between individual application components, such as the software that renders the view or the database that holds and serves the data, individual components can be implemented according to which provides the better solution for the problem at hand.

GeeFu is an application centred around a database schema that holds GFF3 [4] formatted feature data. It has been designed with the needs of researchers wanting to work with high throughput sequencing data from a central place. It is a tool for sharing and storing next-gen data with a minimum of fuss. It is easy to retrieve data via scripts or URL requests, upload data and visualise in a genome browser. In the jargon of web applications it is the Model and Controller of the Model View Controller framework [5]. On request from a web service it will search the database on the basis of the parameters of the request and return a result. As an instance of a Rails application GeeFu can be used with any type of RDBMS currently supported by Rails [1] without modification. At this stage in development GeeFu is capable of receiving and handling requests via a simple interface and can

send data to return AnnoJ [2], a web service based viewing engine for genomic data. It returns data which AnnoJ is able to render.

1.2 Obtaining GeeFu

The current version of GeeFu can be obtained from
`http://github.com/danmaclean/gee_fu/`

2 Installing GeeFu

2.1 Pre-requisites

GeeFu requires the following software and Ruby gem packages to be installed

Software	Version
Ruby	1.8.7*
RDBMS	eg MySQL (version 5.1.39)

*Ruby 1.9.1 does not yet work with the Mongrel webserver we need for the development and tutorial environments, so isn't supported, yet. When this is sorted, we'll be happy to support it. We are also looking at using Rubinius as an alternative implementation of Ruby, but are having some hiccups with our samtools-ruby module. Usage of features solely from the database looks fine with Rubinius.

Ruby gem	Version
rails	2.3.4 or later
RubyGems	1.3.5 or later
bio	1.3.1 or later
json	1.1.9 or later
ffi	0.6.3 or later
mysql*	2.8.1 or later
mongrel	1.1.5 or later

*or whatever the gem is for your RDBMS of choice.

The prerequisites for these will be required too. For an easy life use the RubyGems package manager to install them for you. If you intend to use AnnoJ as a viewer,

then you will need to have an internet connection, some of the components of AnnoJ need to be retrieved on request from the AnnoJ server, it too is a Web Service.

2.2 Installation

GeeFu is just an instance of a Rails application and has no special requirements other than those above. Rails installation is well documented and in most cases is very easy. See the Rails website [1] for full documentation. GeeFu is very easy to install on a local machine for devlopment and setup purposes, for larger production installs see the instructions in the Rails documentation. Copy the gee_fu folder to a place on your hard disk from which you wish to work, we will call this place RAILS_ROOT (because this is what Rails calls it). It need not be in the cgi-bin or equivalent, since Rails comes with its own server for development. See the Rails documentation for further information when it comes to deploying the database or browser you have built into a production environment. Once the gee_fu folder is copied, thats it installed and it is ready to configure.

2.3 samtools-ruby and GeeFu

The Ruby module samtools-ruby, written by Ricardo Gonzalez-Ramirez [**?**] is a Ruby wrapper around the popular and very useful samtools C library [**?**], and is the method by which GeeFu is able to use BAM files as a source of read information. samtools-ruby is in early development and is not yet easy to install for all platforms, Ricardo does plan to have a gem soon. You will need to download samtools-ruby and the libbam source code and compile it for your own platform then make sure it is accessible to GeeFu. The server root (e.g $RAILS\_ROOT$) is a common place for it to go. It is beyond the intent of this document to describe the process of installing and compiling the libbam for multiple systems especially as the problem is likely to go away once samtools-ruby is available as a gem. Hopefully this will be resolved soon. If you do have lots of problems, email us and we'll do our best to help.
The samtools-ruby isn't required for basic functionality of GeeFu, only for BAM use. The tutorial here will work fine even if you can't install libbam and samtools-ruby.

PLEASE NOTE: In the latest update of GeeFu BAM file use isn't working. I seem to have broken it. Ricardo is nearly there with the Gem so we will fix soon. This should be resolved very soon.

2.4 Quick Start

In this section we'll look at setting up and adding some sample data into GeeFu as a way of understanding the basic workflow of GeeFu.

1. Copy the GeeFu directory to somewhere on your disk, this is now $RAILS\_ROOT$
2. Open a terminal or command line and cd into $RAILS\_ROOT$
3. Open the file $RAILS\_ROOT/config/database.yml$ in your text editor. It looks like this.

```
development:
    adapter: mysql
    database: geefu_sample
    username: my_user
    password: ''
    host: localhost
    timeout: 5000
```

   In the development block change the value of the adapter attribute to the type of RDBMS that you have (see the Rails documentation for more info) and change the value of the database attribute to the name you want for your database. Also change the database permissions (username, password and host) as appropriate. The username you specify in database.yml will need create, grant, select and insert privileges on the database specified. The database itself will be created later.

4. In a terminal cd to $RAILS\_ROOT$ and create the database based on the info in database.yml.

```
rake db:create
```

5. Add the schema to the database.

```
rake db:schema:load
```

6. The $max\_allowed\_packet$ variable in your RDBMS may need to be increased to allow for the size of the largest reference sequence in your fasta file, see your RDBMS manual for information on how to do this. For this example data and with MySQL you can do this by executing

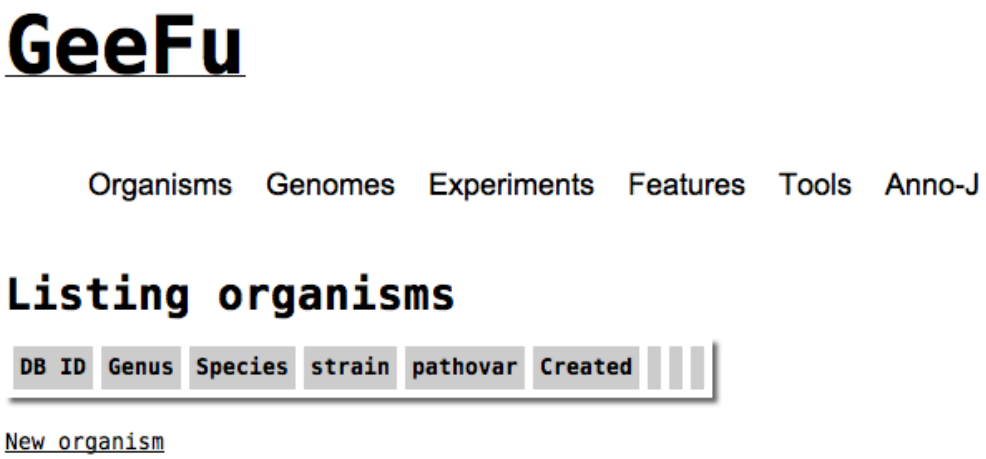
```
set global max_allowed_packet=1000000000;
```

Where the number is a big enough amount of memory in bytes to take your longest sequence. See the information here `http://dev.mysql.com/doc/refman/5.1/en/packet-too-large.html`

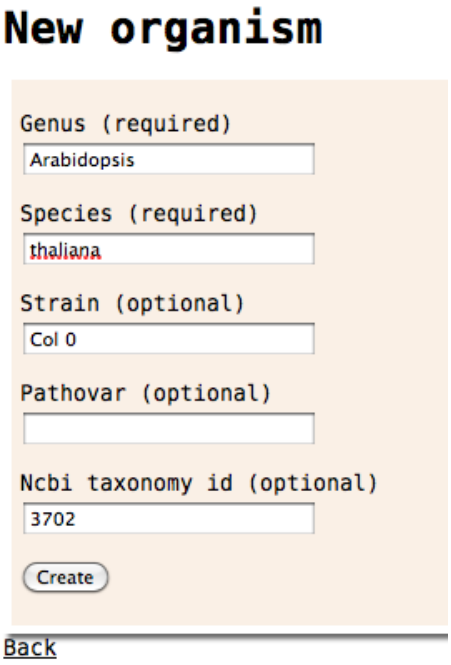
7. Start the built-in Rails webserver on your local machine, from $RAILS\_ROOT$ execute

```
script/server
```

8. Point a web-browser (Firefox, Safari or Chrome will be most compatible with AnnoJ, though others should be fine with just GeeFu) at `http://localhost:3000/begin` You should get a GeeFu page called 'Listing organisms', like below



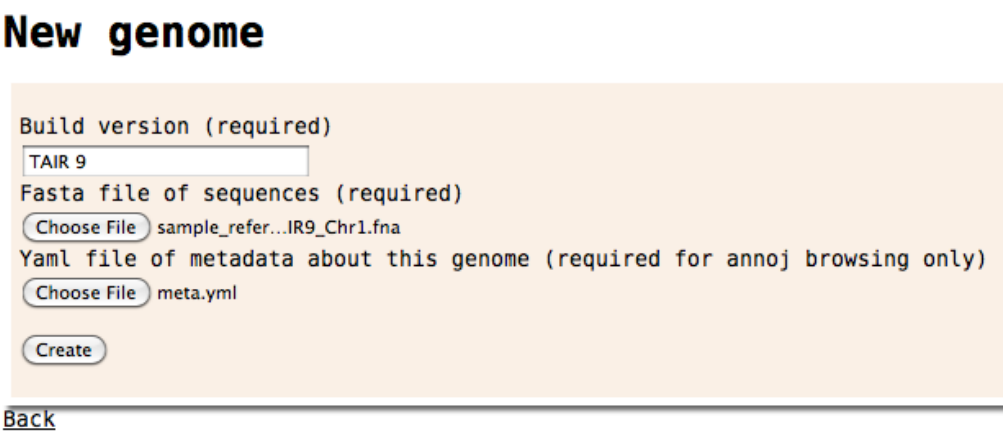
9. Create an Organism, select new organism and fill in the form with the information below.



Click create to save the organism information to the database.

10. Add a Genome Build for this organism. Use the Genomes page link, and then the New Genome link. Fill in the information as below

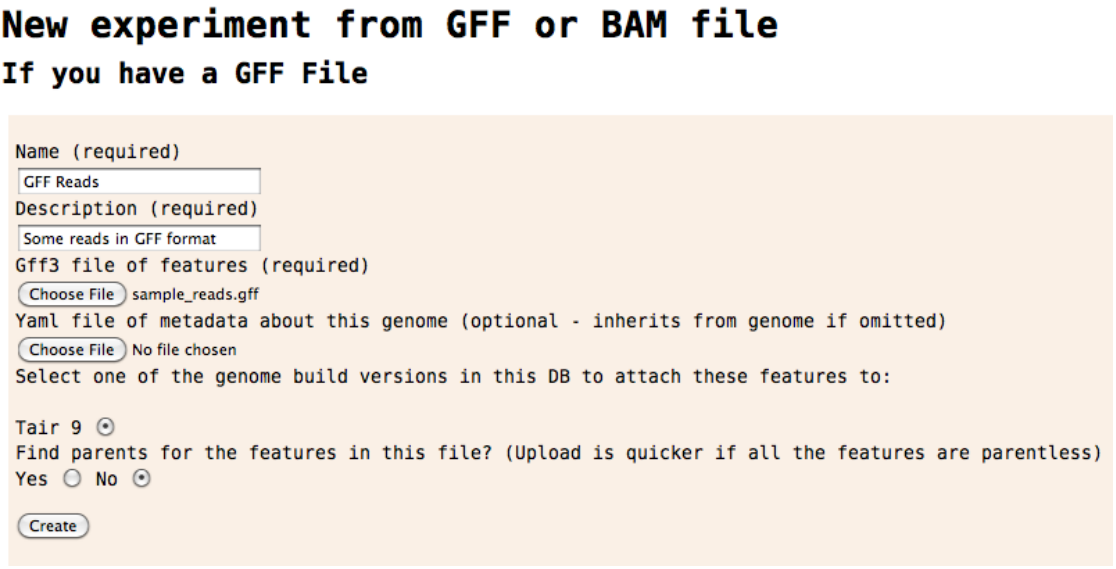
11. Create an Organism, select new organism and fill in the form with the information below.



The fasta file in $public/sequence/sample\_references\_TAIR9\_Chr1.fna$ as the sequence. In the metadata field use the file $public/config/meta.yml$.
If you get an ActiveRecord error here, the most likely explanation is that you forgot to set the max_allowed_packet_variable.
If you make a mistake and the database accepts it you CANNOT edit a genome. You'll have to use the destroy link on the Genomes page and then create a new Genome.

12. Create a new experiment from the sample gene model features files, this experiment will hold the gene models for the sample database and browser. Use the Experiments page link and fill in the top form 'If you have a GFF File The gene models to be provided in the GFF3 file box are in the file $public/sample\_gffs/sample\_features.gff$. The metadata file can be omitted as it will be inherited from the genome. Make sure a genome build version is selected (only one should be visible) and select Yes for Find parents.

13. Create a new experiment from the sample sequence read GFF files.
Now the reads represented as GFF features in the GFF file $public/sample\_gffs/sample\_reads.g$ will be uploaded using the same form, select 'Experiments' and 'New experiment'. Enter the information as below and hit create. This time 'Find Parents' can be 'No', since none of the reads in the file have declared any parents.



14. Create a new experiment from a sample BAM file.
    Again, select 'Experiments' and 'New experiment', this time fill in the bottom form, for a BAM file. Link the database to a bam file containing read alignments in $public/bam/aln.sort.bam$, there isn't a 'Choose File' button to allow browsing as this would try and upload the whole BAM file, we just want the database to know where the file is, so type the file location into the box (The location pointed to may be a symlink to a file situated elsewhere on the disk).

or if you have a BAM File

Name (required)

Sample reads in a BAM file

Description (required)

Reads from a BAM file containir

Bam file of features (required)

public/bam/aln.sort.bam

Yaml file of metadata about this genome (optional - inherits from genome if omitted)

Choose File No file chosen

Select one of the genome build versions in this DB to attach these features to:

Tair 9

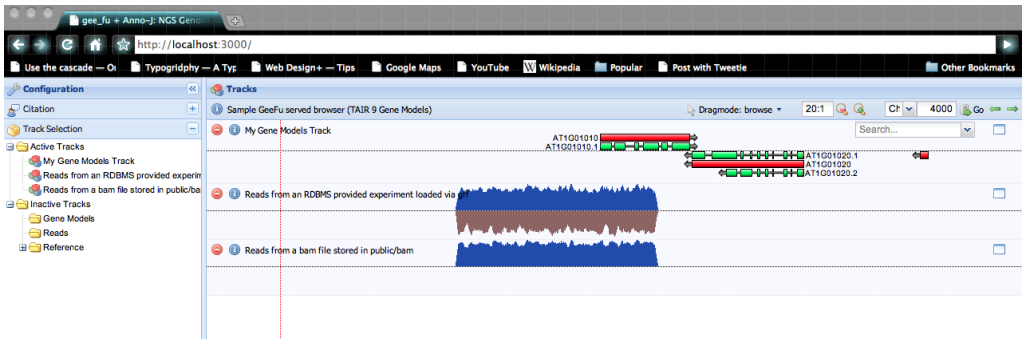
Create

15. Create the file $RAILS\_ROOT/public/javascript/config.js$, from the information in $RAILS\_ROOT/config/config.yml$ using the rake task, go to the terminal and type:

```
1 rake create:config
```

config.js is for AnnoJ and is really just a JSON version of config.yml, so if you make any changes to config.yml, remember to regenerate the config.js file or AnnoJ won't pick them up. AnnoJ uses config.js to set itself up.

16. Click the Anno-J link and you should now get a view of the couple of models and reads as below.
If it didn't work, and you managed to load the sample database without problems, then it may be that you need to change the information in config.yml



Zoom in a bit and the reads tracks should turn into individual reads, eventually with the sequence (you may need to use the resize and scale functions to see the reads and bases). To see the reference sequence you can open the Reference folder on the left and drag the inactive reference track to the Active Tracks above, it will then render in the browser's view.

3 Setting up GeeFu

In this section I will describe in a bit more detail the different stages in setting up GeeFu.

3.1 Setting up the database

3.1.1 Configuring the database with database.yml

You will need an RDBMS and an account with which you can create, insert and select. These accounts are set up as in all Rails apps by editing the $RAILS\_ROOT/config/database.yml$ file. You do not need to create the database, this is done by a Rake task, it mustn't exist already. The top of the file looks like this (for a mysql database), it is in YAML format.

```
1 development:
2   adapter: mysql
3   database: db_name
4   username: my_user
5   password: 'my_pass'
6   host: localhost
7   timeout: 5000
```

change the adapter, database, username and password to suit whatever is appropriate for you. See the Rails documentation for more explanation of what this file is for.

3.1.2 Creating the database

The database is created based on the information in the file database.yml by calling the rake task $rakedb : create$. The schema is added to the database with $rakedb : schema : load$ from $RAILS\_ROOT$, this takes the schema from $db/schema.rb$. If you ever want to alter the structure of the database you should do it with Rails Migrations, not by altering schema.rb or directly altering the database.

3.2 Entering Data

The webforms can all be accessed from the links at the top of each page. The 'home' page is /begin. The three links Organisms, Genomes and Experiments each takes you to a view that will list all the entries of that type currently in the database. At the bottom of the list view is a new link for adding data. Each entry has a Show and Destroy link. Show takes you to a page with all the information about the entry, Destroy lets you remove the entry from the database. Organisms and Experiments can also be edited, that is change the information already in the database for an entry without generating a new database ID.

3.2.1 Adding an organism to the database

Use the Organisms link at the top of each page to see the list of organisms and the New organism link to get the input form.
A GeeFu instance should really only have one organism entry (although it is possible to have any number). The information require is sparse but sufficient to unambiguously define which organism is described in this database, particularly if you provide an NCBI taxonomy id.

Unlike the other datatypes in a GeeFu database, organism isn't linked to any other datatype. Destroying an organism has no effect on anything else.

3.2.2 Adding a new genome to the database

Use the Genomes link at the top of each page to see the list of genome and the New genome link to get the input form.
GeeFu can hold multiple genome build versions for the organism. The webform requires that you provide a Build Version for the genome. This is supposed to indicate the assembly version of the sequences provided in the Fasta file that is also required. When the Fasta file is uploaded, the Reference entries in the database that represent the build/assembly and the related Sequence entries will all be created and linked to this genome.
The metadata file isn't required, but is a good idea. The yaml format allows you to add whatever structured data you like and can be powerful and very descriptive when used well.
Although a yaml file is provided the data is stored internally as a json string.
If you intend to use AnnoJ as a viewer you will need to add some metadata for

that, the required data is described below

When a genome is destroyed, all the associated References and Sequences are destroyed too.

3.2.3 Adding features as an experiment

Use the Experiments link at the top of each page to see the list of Experiments and the New experiment link to get the input form.
The features are stored in the Features table but uploaded as groups termed an experiment. Each experiment can take either a GFF or a BAM file, but not both. An existing Genome must be selected for an Experiment, all the features in that experiment will belong to that experiment and when you destroy the experiment, all the features will be destroyed too. Each experiment can take an individual metadata file in yaml format, if you don't provide one the Genome metadata is used.
Many GFF features have parents, and the parent information is held in a join table in the database, you will never need to access this directly. If you do have parented features in your GFF file, you will need to tell the web form to recognise them. Because of the index, upload is a lot quicker if you can avoid searching for parents when it isn't needed.

The ID attribute is used for parenting. GFF3 format allows for multiple IDs for each feature and multiple parents. In GFF3 the ID must only be unique within the scope of the GFF file. GeeFu does not require that IDs are unique within the database. When multiple IDs are declared GeeFu takes only the first declared ID attribute as the current feature ID (although all IDs are stored as part of the group attributes). When GeeFu searches its database for parents via ID, then all features with that ID, regardless of experiment or genome are marked as parents. This allows such things as contigs in one experiment having supercontigs in another experiment as parents, which is useful for searches via the console/scripts/REST interface, but allows them to stay distinct in terms of metadata and rendering in AnnoJ. Thus it is your responsibility to make sure the GFF IDs are all in order. In practice this isn't so hard, if you stick to the rule of having just one ID per feature and ensuring they are unique in the whole database you should have few problems.

4 Using the GeeFu database

4.1 Editing existing database information

Organism and Experiment data can be modified after entry. Select either Organisms or Experiments and then the edit link for the entry you wish to edit.
The features in an experiment can either be added to by selecting 'merge', which adds the features without replacing features already in the experiment. Selecting 'replace' destroys all existing features and inserts the current ones in place.

4.2 Features Link

This link doesn't do anything yet! It will soon!

4.3 Tools Link

The Tools link shows you the form for web tools built into GeeFu, these are currently:

1. Get some genomic sequence

4.4 Anno-J Link

This will open a new browser window with AnnoJ renderings of the data as defined in config.js.

5 GeeFu and GFF3

GeeFu requires a GFF3 file, comments and directives may be present, but are ignored. GeeFu doesn't do any validation or enforce feature types, this should all be done before hand.
The BioRuby GFF parser that GeeFu uses requires escaping of the following characters in GFF lines, '% = & ,'. These must be escaped with the HTML codes or GFF parsing will fail. Here are some Perl-like regular expressions that will carry out this substitution.

```
$gff =~ s/%/%25;/g;
$gff =~ s/;/%3B;/g;
$gff =~ s/=/%3D;/g;
$gff =~ s/&/%26;/g;
$gff =~ s/,/%2C;/g;
```

The GFF3 format has no specific place to add sequence or quality information for reads. There is a free text Note attribute in the group field though, which can serve this purpose. GeeFu uses two HTML like tags to identify the sequence and quality. Enclose the sequence between <sequence> and </sequence> tags and quality scores between <quality> and </quality> tags. For example

```
Note=<sequence>ATCG<\/sequence><quality >!!!! <\/ quality >;
```

The tags dont have to be used as pairs. When the GFF is added to the database, the tags and information is parsed into separate fields from the rest of the Note attribute and no longer remains within it.

6 Getting data from the database with the RESTful interface

GeeFu can be accessed with a URL GET request and appropriate parameters, just typing the URL into a browser will return data but this is most useful for serving data to web services or web pages or scripts that can access a server via http, this is what makes GeeFu so good for sharing data. Data can be returned in XML or JSON format. All the URLs in this section assume that the base URL is prepended if not shown, eg $www.myserver.ac.uk/specific\_gee\_fu\_stuff$ is just $/specific\_gee\_fu\_stuff$. Typically the syntax for all records in a table is $/table.format$ eg $/genomes.xml$. The syntax for individual records is $/table/id.format$ eg $/genomes/1.xml$. Optional parameters appear after this with the format $?param = value$ and multiple parameters are separated by &. The Rdoc at $RAILS\_ROOT/doc/app/index.html$ describes the available methods and usage. You should look for files called XxxxController, where Xxxx is a table in the database that you are interested in getting information from.

6.1 Getting the list of genomes

Genome information is accessed using the standard RAILS route $/genomes$. The format is specified by the extension $/genomes.xml$ returns XML formatted data $/genomes.json$ returns JSON formatted data.

6.2 Getting a specific genome

Use $/genomes/id.format$ eg $/genomes/1.xml$ for XML data, $/genomes/2.json$ for JSON data. Accessing an ID that doesn't exist will return an empty result.

6.3 Getting the list of experiments

Experiment information (not features) is accessed using the standard RAILS route $/experiments$. eg $/experiments.xml$ or $/experiments.json$.

6.4 Getting a specific experiment

Use $/experiment/id.format$ eg $/experiments/1.xml$ for XML data, $/experiment/2.json$ for JSON data. Accessing an ID that doesn't exist will return an empty result.

6.5 Getting the list of references for all genomes

Reference information (not sequence) is accessed using the standard RAILS route $/reference$. eg $/references.xml$ or $/references.json$.

6.6 Getting the list of references for a specific genome

Use $/references/genome\_id.format$ eg $/references/1.xml$ for a list of references for genome 1 as XML data, Accessing an ID that doesn't exist will return an empty result.

6.7 Getting a specific reference by genome id and name

Use the name parameter after the genome id
ie $/references/genome\_id.format?name = reference\_name$. For example, $/references/1.xml?nam$
$reference_name$

6.8 Getting a reference's sequence

The general parameter for returning the sequence for the reference is sequence=true as a GET parameter, eg $/references/1.xml?sequence = true$

6.9 Getting Feature information

It isn't usually feasible to get a list of features, rather features are returned in answer to a request that sets limits on the genome, experiment, reference, start and end. Entire objects or depth over the range can be returned.

6.9.1 Getting all feature objects within a start and end point on a reference in an experiment

Use $/features/objects/experiment\_id?parameters$. Required params are reference=ref_name, start=start_nucleotide, end=end_nucleotide, eg /features/objects/1?reference=Chr1&start=1&end=10000. Optional parameters: format=json or xml (default = xml), sets the format for return data overlap=true or false (default = false) if true objects that overlap the limits will be returned and not just those completely contained within. If the data is coming from a BAM file the behaviour of samtools-ruby is currently such that the objects returned will always be selected as if overlap=true. type=some GFF feature type, only objects with GFF feature type will be returned.

6.9.2 Getting a summary of coverage information between a start and end point on a reference in an experiment

Use $/features/depth/experiment\_id?parameters$. Required parameters are reference=ref_name, start=start_nucleotide, end=end_nucleotide,
eg $/features/depth/2?start = 10$&$end = 1080$&$format = json$&$reference = Chr1$
Optional Parameters: overlap=true or false (default = false) if true objects that overlap the limits will be returned and not just those completely contained within. If the data is coming from a BAM file the behaviour of samtools-ruby is currently such that the objects returned will always be selected as if overlap=true.

The hash summarising depth in the region selected is quite complex, roughly it is position => strand => nucleotide = count of nucleotides in JSON it looks like this

```
{"some_nucleotide_position":
  {
  "+":
    {"A":0,
     "T":0,
     "G":0,
     "C":0,
     "N":0,
     "strand_total":0
     },
  "-":
    {"A":0,
     "T":0,
     "G":0,
     "C":0,
     "N":0,
     "strand_total":0
     },

  "position_total":0
  },
  ...
  "region_total":0
}
```

For every covered nucleotide position there is a hash with keys for + and - strand and total read bases covering that position (position_total). Each strand has a hash that has keys for each of A,T,C,G or N and total read bases covering that position on the strand (strand_total). The whole thing has a key called region total that counts the total number of read bases in the entire region.

7 Getting data from the database with the Rails console

Rails has an interactive console interface with which you can use to access the data and objects using the Models and methods defined in GeeFu. An enthusiastic introduction can be seen here `http://slash7.com/2006/12/21/secrets-of-the-rails-console-ninjas/`. The syntax is pure Ruby and is very straightforward. The models from GeeFu that are already loaded are the tables of the database and the standard methods cred-

ited to the objects are available. Some useful ones include the $Feature.find\_in\_range$ and $Feature.find\_in\_range\_no\_overlap$, (the Rdoc for the Feature class in $RAILS\_ROOT/doc/app/index.html$ describes each of the methods.

8 Getting data from the database with Ruby scripts

Interacting with the GeeFu database directly isn't a good idea, it is very easy to use the Rails environment and take advantage of the objects and methods created by using either the interactive console, which can be started by typing $script/console$ from within $RAILS\_ROOT$ or via the $script/runner < my_ruby_script.rb >$ which will load the GeeFu environment and run your script on it.

9 Using AnnoJ as a view port

AnnoJ is a "Anno-J is a Web 2.0 application designed for visualizing deep sequencing data and other genome annotation data". AnnoJ does require that some sort of service send it the data for it to render in the first place, GeeFu is great for this and can use AnnoJ as a fairly straightforward way to visualise the data it holds. GeeFu is set up to enable interaction with this view port, but can be easily modified to work with others.

9.1 Requirements

9.1.1 Browser requirements

AnnoJ insists on W3C compliant browsers, Internet Explorer doesn't hack it. Right now, Firefox 3.6 is giving some rendering issues though Firefox 3.5 seems fine. The latest versions of Chrome and Safari on my Mac seem fine too.

9.1.2 Connection requirements

AnnoJ is a web-service browser and the idea is that all of its components are retrieved from over the internet. GeeFu carries most of the components of AnnoJ to make GeeFu as stable as possible but some can't be retrieved or kept locally so an internet connection is required.

9.2 Configuring AnnoJ for use with GeeFu

9.2.1 config.js and config.yml

The AnnoJ view requires a JSON format configuration file, to tell it about the genome and the service etc. JSON is a data-serialisation format that is fairly easy to write but there are a lot of brackets and it can be a pain to keep up. YAML is easier so GeeFu allows you to maintain a YAML file and convert it to the $RAILS\_ROOT/public/javascripts/config.js$ AnnoJ uses. Edit the file $RAILS\_ROOT/config/config.yml$. The most important are the tracks stanzas, this is where you tell AnnoJ the tracks that should be available in your browser. As far as I can tell from the undocumented Annoj it looks like there are 3 track types: ModelsTrack, for aggregated transcript style gene models; RepeatsTrack, for strandless block objects like repeats and ReadsTrack, for high throughput sequencing reads that will be be shown at various zoom-levels and the reference sequence GeeFu provides that is sent to AnnoJ as if it is a big read. Here are the different configuration options for the rest

```
1  -id:     #a numeric unique id for the track
2   name:   #free text name
3   type:   #one of ModelsTrack, RepeatsTrack, ReadsTrack
4   path:   #the place in the track selection panel where this track
        appears
5   data: /features/method/x # method =
6             # 'annoj' for feature (models, reads and other GFF stuff)
7             # OR 'chromosome' for the reference sequence track
8             # x =
9             # experiment ID or genome ID from the GeeFu database
10  showControls:  # true to see the individual lane control button or
        absent
11  height: 80 # start up height of the  track in pixels
12  minHeight: 20  # height of track when minimised
13  maxHeight: 60  # height of track when maximised
14  single:    # true if the track is strandless otherwise absent
```

The active stanza lists the tracks that will be shown when the page loads, use the id of the track. The genomes bit is the url of the service that provides information about the genome, use genomes/x, where x is the id of the genome in the genomes table of GeeFu. Usually 1. The bookmarks and stylesheets stanzas seem redundant (perhaps they are for unimplemented features of AnnoJ), leave them. Location is where to focus AnnoJ on start up. Bases and pixels refer to the zoom level at startup. Its Bases:pixels. Admin is the name and contact details that appear in the information panel in AnnoJ in the information panel for the whole

browser.
Once the config.yml file is made, you can turn it into the config.js needed with the rake task rake create:config, which you execute from within RAILS_ROOT.

9.2.2 meta.yml

AnnoJ requires certain metadata to 'syndicate'. That is to understand what it is that it is supposed to be rendering. This is required at a genome and individual track level within AnnoJ. When you create an experiment or genome you have the option to add a yml metadata file. As a minimum you should include the following information for your genome:

```
---
institution:
  name: Your Institute
  logo: images/institute_logo.png
  url: http://www.your_place.etc
service:
  format: Unspecified
  title: Sample GeeFu served browser (TAIR 9 Gene Models)
  server: Unspecified
  version: Vers 1
  access: public
  description: Free text description of the service
engineer:
  name: Mick Jagger
  email: street.fighting.man@stones.net
genome:
  version: TAIR9
  description: Chr1 from TAIR9
  species: Arabidopsis thaliana
```

The values can be whatever you like, as long as all this information is present. You only need to upload this information for the genome. If you choose not to provide specific information for an experiment, it will get the same information as its parent genome when it is created.

9.3 Accessing the view in a web browser

The configured GeeFu/AnnoJ instance is accessed via the index.html which resides in the $RAILS\_ROOT/publicfolder$. Point your browser at this to start the view. See Rails documentation for more information. If you used the built in server

for development try localhost:3000 otherwise the AnnoJ view will be wherever your sys-admin installed the server for you.

9.4 GFF and AnnoJ

Although GeeFu doesn't take exception to any feature type, AnnoJ's support for complicated aggregated features like transcripts isn't well documented and in my experience sometimes some feature types render over others or not at all. For an easy life with model tracks in AnnoJ use genes and mRNA, five_prime_utr, three_prime_utr and CDS and exon in the gff file that specifies the models. If you want to add other features, such as polypeptides or ESTs add them into tracks of their own. As a workaround to this behaviour GeeFu will only attempt to aggregation of the following sub-feature types 'intron', 'exon', 'CDS', 'five_prime_UTR', 'three_prime_UTR', 'start_codon', 'stop_codon', 'match_part' , any other feature types won't aggregate. You can alter this by altering the list in the to_annoj method in $/RAILS\_ROOT/app/models/feature.rb$. We may make this a configurable list in the future.

9.5 Changing feature rendering in AnnoJ

The colour and fill of feature objects in the view are determined in the $RAILS\_ROOT/public/stylesheets/plugins.css$ file. The class of the element refers to the GFF feature type, so $.mRNA\{someCSS\}$ is what defines the look of mRNA features. The images can be changed if you wish and are stored in RAILS_ROOT/public/images. If you want different tracks with the same feature type to have different appearances you are out of luck. To achieve this you will have to change feature types on one of the tracks.

10 Working with GeeFu at the 'back-end'

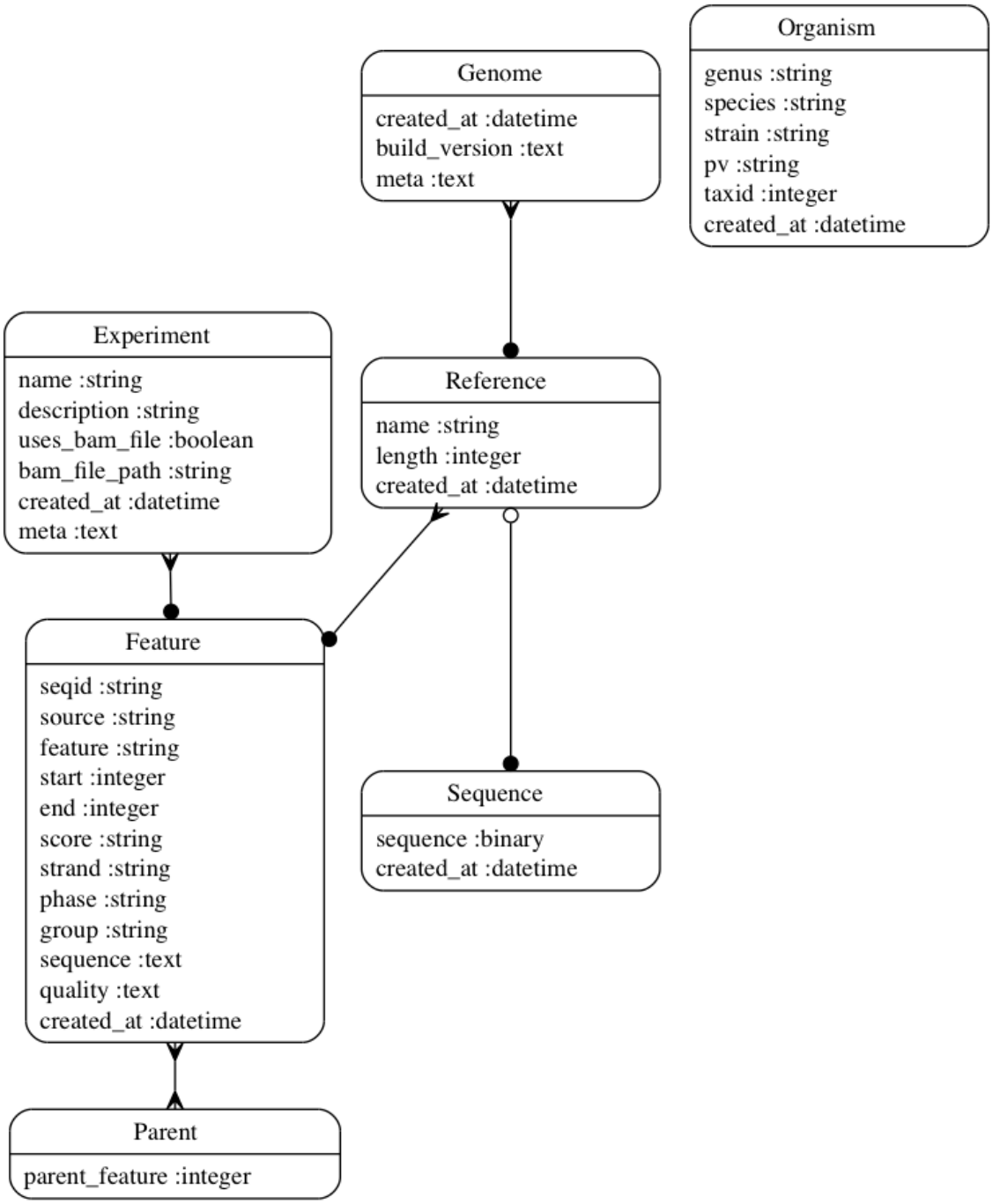
10.1 Rake Tasks

Rake tasks are Rails built in way of automating certain processes during development of the database. from RAILS_ROOT you can execute $rake --tasks$ to get a full list. The majority in there are the standard Rails ones, but here are the ones added specifically for GeeFu.

add:species	add an organism to the database
add:sequences	add Fasta sequences into the database
add:reference_info	add reference entries
add:features	add features in an experiment (track) to the database
remove:features	remove features in an experiment (track) from the database
list:experiments	lists the experiments added to the database already
create:config	creates the AnnoJ config.js file based on the information in config.yml

10.2 Schema

The schema is very straightforward and easily extended. It consists of a central Features table and a many to many join table Parents that indicates which features are parents (according to their gff records) of which other features. References all belong to a genome, as do Experiments and features all belong to an experiment. Sequences belong only to a reference. Organisms is a seperate table that has no relationship with anything else and serves mostly to identify the organism whose sequence and features are under study


Organism
genus :string
species :string
strain :string
pv :string
taxid :integer
created_at :datetime
Genome
created_at :datetime
build_version :text
meta :text
Experiment
name :string
description :string
uses_bam_file :boolean
bam_file_path :string
created_at :datetime
meta :text
Reference
name :string
length :integer
created_at :datetime
Feature
seqid :string
source :string
feature :string
start :integer
end :integer
score :string
strand :string
phase :string
group :string
sequence :text
quality :text
created_at :datetime
Sequence
sequence :binary
created_at :datetime
Parent
parent_feature :integer

10.3 The tools page

We provide a model-less controller as a place to hold web tools that act on the database. Currently we have a genome sequence extraction tool which renders a web-form and retrieves sequence as per the information specified by the user in the form using the code $/tools$. The web form itself is specified in $RAILS\_ROOT/app/views/index.html.erb$ and the code it calls is the method $genomic\_sequence$ in $RAILS\_ROOT/app/controllers/tools\_controller.rb$. The output goes to the output webpage $/RAILS\_ROOT/app/views/genomic\_sequence.html.erb$ in a Ruby instance variable and is interpreted by the erb (embedded Ruby) in that file.

10.4 Extending the database and creating new functionality

You can extend the database exactly as if it were any other Rails application. Adding new tables and datatypes is done easily, just add these as if they were new Rails models. A good way is using Rails migrations to build your tables (that is, get Rails to do the heavy lifting in the Sql side and version your database for you) then use Rails lovely magic convention over configuration to create all the code and models. This is really the start of building Rails apps so see the Rails documentation. See the Rails documentation for conventions for creating and naming new tables, Rails prefers convention over configuration so you should pay attention to these. Adding new functionality to the web app is covered by the same documentation. Briefly, the workflow is to create some web form for collecting input, that is accessible at some url, like the tools index.html.erb. The form sends information to the controller and action specified, like $RAILS\_ROOT/app/controllers/tools\_controller.rb$ and $genomic\_sequence$. The controller and action use the information from the form to do some work on a database, put the result into instance variables and call the view that matches the action like $RAILS\_ROOT/app/views/tools/genomic\_sequence.html.erb$. The view knows how to render the data in the instance variables.

11 Tips for a moving GeeFu to a production server

Once you've got comfortable using GeeFu and want to move to a production server the web-server you choose can be very important in determining the performance of the application. We have had excellent results from Phusion Passenger `http://www.modrails.com/`

12 And Finally...

If you get really frustrated with the software, feel free to complain to me dan.maclean@tsl.ac.uk. I freely admit its early days for this project so it is likely to wobble, however, a lot of your initial problems will be answered in the Rails community pages, please look there if your problem looks like it might be related to Rails more directly than this particular instance of a Rails app.

References

[1] Ruby on rails. http://rubyonrails.org/.

[2] Annoj. http://www.annoj.org/.

[3] Wikipedia web service article. http://en.wikipedia.org/wiki/Web_service.

[4] The gff3 standard. http://song.sourceforge.net/gff3.shtml.

[5] Wikipedia mvc article. http://en.wikipedia.org/wiki/Model-view-controller/.

[6] samtools-ruby at github. http://github.com/homonecloco/samtools-ruby.

[7] samtools library. http://samtools.sourceforge.net/.