pyucsc Documentation

Release 0.2

James Casbon

February 07, 2017

Contents

CHAPTER 1

Overview

UCSC Genome Bioinformatics provides sql tables and fasta files for many genomes. pyucsc provides a lightweight python interface to these resources that can provide an SQL interface (using SqlAlchemy) and fast access to the DNA sequences. It ties these together with a set of model objects that are loaded from the database and can be directly interrogated for the DNA sequence. All genomic intervals are described using fastinterval which provides convenient interval operations and fast sequence loading via pyfasta.

1.1 Choosing a genome

We first need to create an SqlAlchemy *Session* and fastinterval *Genome*. This requires that you have configured the database server and directory to use (see *Configuration*):

```
>>> import ucsc
>>> session, genome = ucsc.use('hg19')
```

1.2 DNA intervals and Fasta files

The core of the interface to the sequence is the *fastinterval.Interval* class. All genomic locations are described with this class:

```
>>> i1 = genome.Interval(10182300,10182320, chrom='chr3')
>>> i1.sequence
'agctcactgcaacctccgcc:'
>>> str(i1)
'chr3:10182300-10182320:'
```

Strands are handled and reverse complements generated correctly

```
>>> i2 = genome.Interval(10182300, 10182320, chrom='chr3', strand=-1)
>>> i2.sequence
'ggcggaggttgcagtgagct'
```

Interval logic is available as methods on the Interval class

```
>>> i3 = Interval(10182310,10182330, chrom='chr3')
>>> i1.overlaps(i3)
True
>>> i1.contains(i3)
False
>>> i1.intersection(i3)
```

```
Interval(10182310, 10182320)
>>> i1.span(i3)
Interval(10182300, 10182330)
>>> i1.union(i3)
Interval(10182300, 10182330)
```

For full details, see the fastinterval documentation.

1.3 SQL tables

pyucsc uses SqlAlchemy to expose the ucsc database tables:

```
>>> from ucsc import tables
```

e.g. to count the entries in knownGene:

```
>>> tables.knownGene.count().execute().scalar()
77614L
```

or to get genes:

```
>>> tables.knownGene.select().limit(2).execute().fetchall()
[('uc001aaa.3', 'chr1', '+', ...),
 ('uc010nxq.1', 'chr1', '+', ...)]
```

See the SqlAlchemy SQL core tutorial for more information.

The foreign key relationships are not defined by UCSC in the schema. This means that, for the moment, you must manually specify conditions when constructing a join.

1.4 Model objects

Alternatively, we can use the pyucsc model objects which provide a more natural python interface to the database tables, with convenience methods such as creating appropriate intervals:

```
>>> from ucsc import model
>>> vhl = session.query(model.KnownGene).filter(model.KnownGene.geneSymbol=='VHL').one()
>>> vhl
KnownGene(VHL, uc003bvc.2, chr3:10183318-10193744)
```

To get the transcript:

```
>>> vhl.transcript
Interval(10183318, 10193744)
>>> vhl.transcript.sequence
'CCTCGCCTCCGTTACAACGGCCTACGGTGCTG...'
```

Snp queries:

```
>>> model.Snp.for_interval(vhl.transcript)
[SNP(rs779805, chr3:10183336-10183337),
 SNP(rs34271731, chr3:10183434-10183435),
 ... ]
```

For the full model documentation, see *Table Models*.

1.5 Configuration

You need to provide a local copy of the Fasta files and the database server to use. You *can* use the public UCSC mysql server, but please respect their usage policy. Configuration is achieved either through a configuration file, or by setting variables in *ucsc.config*.

Via Configuration Files

Create a YAML file in either */etc/pyucsc* or *~/.pyucsc* with two entries:

```
fasta_dir: /fasta
database_uri:  mysql://genome:@datarig.local/
```

Via code:

```
from ucsc import config
config.fasta_dir = "/fasta"
config.database_uri = "mysql://genome:@datarig.local/"
```

1.6 Status

The table interfaces are generated by introspection and therefore complete. The model interface only covers a limited set of tables, but it is easy to add new classes and mappings.

1.7 Development

Please use the github repository for issues and patches: https://github.com/PopulationGenetics/pyucsc

CHAPTER 2

Table Models

The model objects are automatically loaded from the database and populated with the attributes from the table. A *KnownGene* object will therefore have a *txStart*, *txEnd*, etc attributes. The mapping from database tables to objects is performed by SqlAlchemy for us but to query the tables you need to use the SqlAlchemy ORM interface

Below we list the model methods we have added to the basic data. To find the attributes belonging to each class, you can use the UCSC table browser's *describe table schema* button on the Table Browser

class ucsc.model.**CcdsGene**
Bases: *ucsc.model.KnownGene*

ccdsGene entry, same interface as KnownGene

cds
return an Interval representing the CDS

exons
return a list of Intervals for each exon

transcript
return an Interval representing the transcript

class ucsc.model.**ChainSelf**
Bases: ucsc.model.QueryByInterval

Chainself entry

dest
Interval of the destination of the chain

for_interval(*interval*)
return all links that overlap the specified interval

source
Interval of the source of the chain

class ucsc.model.**ChainSelfLink**
Bases: ucsc.model.QueryByInterval

ChainSelfLink entry

dest
Interval of the destination of the chain

for_interval(*interval*)
return all links that overlap the specified interval

source
Interval of the source of the chain

class ucsc.model.**CommonSnp**
Bases: *ucsc.model.Snp*

SNP entry

apply(*interval*)
Create the alernate alleles on the given interval

returns a list of alleles over the interval given

for_interval(*interval*)
Return all snps within an interval

interval
Get this Snp's interval

other_alleles()
return the alternate allele (always on the + strand, unlike observed)

class ucsc.model.**KnownCanonical**
Bases: *ucsc.model.KnownGene*

canonical genes

cds
return an Interval representing the CDS

exons
return a list of Intervals for each exon

transcript
return an Interval representing the transcript

class ucsc.model.**KnownGene**
Bases: object

knownGene entry

cds
return an Interval representing the CDS

exons
return a list of Intervals for each exon

transcript
return an Interval representing the transcript

class ucsc.model.**RefGene**
Bases: *ucsc.model.KnownGene*

refGene entry, same interface as KnownGene

cds
return an Interval representing the CDS

exons
return a list of Intervals for each exon

transcript
return an Interval representing the transcript

class ucsc.model.**Snp**

Bases: object

SNP entry

apply(*interval*)

Create the alernate alleles on the given interval

returns a list of alleles over the interval given

classmethod for_interval(*interval*)

Return all snps within an interval

interval

Get this Snp's interval

other_alleles()

return the alternate allele (always on the + strand, unlike observed)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

u