Introducing GPy (Genomic Python)

A framework for genomic analysis
Motivation

- Tired of Perl scripts with little code-reusability
- Tired of managing innumerable sequence files
- No uniform way to 'objectize' genome
- Ensembl API exists, but…
  - not really object oriented
  - limited to Ensembl data
- Genomic data representation in RDMS unnatural and anti-object oriented
- SQL queries somewhat restricted
Why Python?

- Everything is object-oriented (including Python)
- Python is interpreted and interactive
- Python combines remarkable power with exceptionally clear syntax
- Python is portable
- People (who use Python) LOVE Python
Overview

- GPy Concepts
  - Features Galore
- Extending GPy
  - Featuring UCSC Genome Browser Database
- Case study: Results from SynapseDB
  - Querying microarray data
- Future Direction
  - ZODB
  - ZOPE
GPy concepts

- Feature...
  - is a named container on a chromosome
  - can hold other Features
  - is the functional unit of genomic representation in GPy
- GPy has Features Galore
  - (Almost) Everything is a Feature

Name: 'HelloGenome'
Chr: 'X'
Start: 1000
Stop: 2000
Feature Instantiation

- **Feature constructor**
  - `Feature(name, chr, start, stop, sense='+')`

```python
>>> from GPy import *
>>> a = Feature('A', 'X', 100, 200)
>>> b = Feature('B', 'X', 300, 450)
>>> c = Feature('C', 'X', 350, 500)
>>> print a, b, c
A->chrX:101-200 B->chrX:301-450 C->chrX:251-500
```
Features contain other Features

- Backpropogation allows Super-Features to span sub-Features

```python
>>> a.append(b)
>>> print a
(A,B)->chrX:101-450
>>> print b
B->chrX:301-450
```
Features interact with each other

- Positional queries
  - `Feature.spans(otherFeature)`
  - `Feature.touches(otherFeature)`

- Hierarchical queries
  - `Feature in Feature`

```python
>>> print a.spans(b)
True
>>> print b in a
True
>>> print a.touches(c)
True
```
Constructing a Feature Heirachy

Name: '(A,(B,C))'
  Chr: 'X'
  Start: 100
  Stop: 500

Name: 'A'
  Chr: 'X'
  Start: 100
  Stop: 200

Name: 'B'
  Chr: 'X'
  Start: 300
  Stop: 450

Name: 'C'
  Chr: 'X'
  Start: 350
  Stop: 500

Name: '(B,C)'
  Chr: 'X'
  Start: 300
  Stop: 500

>>> a.append(b)
>>> print a
(A,B)->chrX:101-450

>>> b.append(c)
>>> print b
(B,C)->chrX:301-500

>>> print a
(A,(B,C))->chrX:101-500
**Featureing Genome Browser in GPy**

- UCSC Genome Browser is the convergence of a wealth of genomic information
  - Multiple species and sequences
  - Gene annotations
  - Comparative genomics
  - Expression and regulation
  - Variation and repeats…
- Backend is many MySQL relational databases
- **ucsc** extends GPy to **Feature GB** databases
ucsc Concepts

- **GenomeBrowserDB objects**
  - Generate **Features** based on a given database
  - Generate **_View**s of genomic sequence
- **_View objects**
  - Inherit from **Feature**
  - Enable group operations
    - _View.addFeaturesFromTrack(track)
  - Enable group queries on sub-**Features**
    - _View.getSequence(feature)
    - _View.getFirstExons()
    - _View.getUtrs()...
>>> from ucsc import *
>>> db = GenomeBrowserDb('hg17')
>>> bdnf = db.getFeaturesBySym('BDNF')
>>> print bdnf
BDNF->chr11:27633020-27699872
Exploring Gene Hierarchy

>>> for transcript in bdnf:
    print transcript
    ... for exon in transcript:
    ...    print "\t", exon
    ...
NM_170735->chr11:27633018-27637249
    NM_170735.0->chr11:27633018-27637249
    NM_170734->chr11:27633018-27677756
    NM_170734.0->chr11:27677505-27677756
    NM_170734.1->chr11:27633018-27636708
NM_001709->chr11:27633018-27678552...
Alternatively…

```python
>>> bdnf.printTree()
BDNF->chr11:27633018-27699872
   NM_170735->chr11:27633018-27637249
      NM_170735.0->chr11:27633018-27637249
   NM_170734->chr11:27633018-27677756
      NM_170734.0->chr11:27677505-27677756
      NM_170734.1->chr11:27633018-27636708
   NM_001709->chr11:27633018-27678552
      NM_001709.0->chr11:27678287-27678552
      NM_001709.1->chr11:27633018-27636708...
```
Querying Sequence

- Features alone are independent of database
- `_View from db` required to query sequence

```python
>>> bdnfView = db.getView('BDNF', \
... bdnf.getChr(), bdnf.getStart(), bdnf.getStop())
>>> bdnfView.append(bdnf)
>>> print bdnf.getSeq()
Seq('atgtactttgaaaatatatttaaaaacattaaaaattctatatttaaaaca
tatattata ...', IUPACUnambiguousDNA())
```

- `getSeq()` returns Biopython `Seq`
Querying Expression

```python
>>> bdnfView.addFeatureFromTrack ('gnfAtlas2')
>>> probes = bdnfView.getTrackFeature('gnfAtlas2')
>>> print probes
206382_s_at->chr11:18-42914
>>> print probes[0].getName2Ratio()
{'testis_interstitial': 0.68899999999999995, 'heart': -1.274, ...}
```
Gene Family Case Study

- In large gene families from SynapseDB how correlated is protein similarity to differences in expression?

```python
db = GenomeBrowserDb('hg17')
geneFamilySyms = ['Rab3a', 'Rab3b', 'Rab3c', 'Rab3d', 'Rab5a', ...
genes2probes = {}
for sym in geneFamilySyms:
    gene = queryGene(db, sym)
    probe = queryProbe(db, sym)
    if gene and probe:
        genes2probes[gene] = probe
dnaDistanceMatrix = getDnaDistanceMatrix(genes2probes)
expDistanceMatrix = getExpDistanceMatrix(genes2probes)
print mantelTest(dnaDistanceMatrix, expDistanceMatrix)
```
How similar is protein similarity to expression difference?

Rab cDNA vs Expression

P = 0.09

N = 2000   Bandwidth = 0.02887
Results

- Rab3a (human)
- Rab3a (mouse)
- Rab3c (mouse)
- Rab3c (human)
- Rab3b (human)
- Rab3b (mouse)
- Rab3d (human)
- Rab3d (mouse)
- Rab5c (mouse)
- Rab5c (human)
- Rab5a (human)
- Rab5a (mouse)
- Rab5b (human)
- Rab5b (mouse)
ZODB and ZOPE

- ZODB (ZOPE Object Database)
  - Powerful object database for Python objects
  - Features transparency
    - Behaves just like a Python dictionary
- ZOPE
  - Python open-source web application server
  - Used to build content management systems on ZODB
Future Directions

- Extend **ucsc** to **Feature all data**
- Optimize queries
- Use Zope and ZODB to store data
- Publish as open-source and let evolve