Python client library

Getting started

The first step is to import the module and initialize the CellBaseClient:

```
>>> from pycellbase.cbclient import CellBaseClient
>>> cbc = CellBaseClient()
```

The second step is to create the specific client for the data we want to query (in this example we want to obtain information for a gene):

```
>>> gc = cbc.get_gene_client()
```

And now, you can start asking to the CellBase RESTful service by providing a query ID:

```
>>> tfbs_responses = gc.get_tfbs('BRCA1')  # Obtaining TFBSs for BRCA1 gene
```

Responses are retrieved as JSON formatted data. Therefore, fields can be queried by key:

```
>>> tfbs_responses = gc.get_tfbs('BRCA1')
>>> tfbs_responses[0]['result'][0]['tfName']
'E2F4'

>>> transcript_responses = gc.get_transcript('BRCA1')
>>> 'Number of transcripts: %d' % (len(transcript_responses[0]['result']))
'Number of transcripts: 27'

>>> for tfbs_response in gc.get_tfbs('BRCA1,BRCA2,LDLR'):
...     print('Number of TFBS for "%s": %d' % (tfbs_response['id'],
len(tfbs_response['result'])))
'Number of TFBS for "BRCA1": 175'
'Number of TFBS for "BRCA2": 43'
'Number of TFBS for "LDLR": 141'
```

Data can be accessed specifying comma-separated IDs or a list of IDs:

```
>>> tfbs_responses = gc.get_tfbs('BRCA1')
>>> len(tfbs_responses)
1

>>> tfbs_responses = gc.get_tfbs('BRCA1,BRCA2')
>>> len(tfbs_responses)
2

>>> tfbs_responses = gc.get_tfbs(['BRCA1', 'BRCA2'])
>>> len(tfbs_responses)
2
```

If there is an available resource, but there is not an available method in this python package, the CellBaseClient can be used to create the URL of interest and query the RESTful service:

```
>>> tfbs_responses = cbc.get(category='feature', subcategory='gene',
query_id='BRCA1', resource='tfbs')
>>> tfbs_responses[0]['result'][0]['tfName']
'E2F4'
```

Optional filters and extra options can be added as key-value parameters (value can be a comma-separated string or a list):

```
>>> tfbs_responses = gc.get_tfbs('BRCA1')
>>> len(res[0]['result'])
175

>>> tfbs_responses = gc.get_tfbs('BRCA1', include='name,id')
>>> len(res[0]['result'])
175

>>> tfbs_responses = gc.get_tfbs('BRCA1', include = ['name', 'id'])
>>> len(res[0]['result'])
175

>>> tfbs_responses = gc.get_tfbs('BRCA1', limit=100)
>>> len(res[0]['result'])
100

>>> tfbs_responses = gc.get_tfbs('BRCA1', skip=100)
>>> len(res[0]['result'])
75
```

What can I ask for?

The best way to know which data can be retrieved for each client is either checking out the RESTful web services section of the CellBase Wiki or

the CellBase web services

If we do not know which method is the most adequate for our task, we can get helpful information for each data-specific client:

```
>>> cbc.get_region_client().get_help()
RegionClient
    - get_clinical: Retrieves all the clinical variants
    - get_conservation: Retrieves all the conservation scores
    - get_gene: Retrieves all the gene objects for the regions. If query
param histogram=true, frequency values per genomic interval will be
returned instead.
    - get_model: Get JSON specification of Variant data model
    - get_regulatory: Retrieves all regulatory elements in a region
    - get_repeat: Retrieves all repeats for the regions
    - get_sequence: Retrieves genomic sequence
    - get_tfbs: Retrieves all transcription factor binding site objects for
the regions. If query param histogram=true, frequency values per genomic
interval will be returned instead.
    - get_transcript: Retrieves all transcript objects for the regions
    - get_variation: Retrieves all the variant objects for the regions. If
query param histogram=true, frequency values per genomic interval will be
returned instead.
```

We can get the accepted parameters and filters for a specific method of interest by using the *get_help* method:

```
>>> cbc.get_region_client().get_help('get_gene', show_params=True)
```

Configuration

Configuration stores the REST services host, API version and species.

Getting the default configuration:

```
>>> ConfigClient().get_default_configuration()
{'version': 'v4', 'species': 'hsapiens', 'rest': {'hosts':
['http://bioinfo.hpc.cam.ac.uk:80/cellbase']}}
```

Showing the configuration parameters being used at the moment:

```
>>> cbc.show_configuration()
{'host': 'bioinfo.hpc.cam.ac.uk:80/cellbase', 'version': 'v4', 'species':
'hsapiens'}
```

A custom configuration can be passed to CellBaseClient with a ConfigClient object. JSON and YML files are supported:

```
>>> from pycellbase.cbconfig import ConfigClient
>>> from pycellbase.cbclient import CellBaseClient

>>> cc = ConfigClient('config.json')
>>> cbc = CellBaseClient(cc)
```

A custom configuration can also be passed as a dictionary:

```
>>> from pycellbase.cbconfig import ConfigClient
>>> from pycellbase.cbclient import CellBaseClient

>>> custom_config = {'rest': {'hosts':
['bioinfo.hpc.cam.ac.uk:80/cellbase']}, 'version': 'v4', 'species':
'hsapiens'}
>>> cc = ConfigClient(custom_config)
>>> cbc = CellBaseClient(cc)
```

If you want to change the configuration on the fly you can directly modify the ConfigClient object:

```
>>> cc = ConfigClient()
>>> cbc = CellBaseClient(cc)

>>> cbc.get_config()['version']
'v4'

>>> cc.version = 'v3'
>>> cbc.get_config()['version']
'v3'
```

Script example

```
# Loading CellBase and configuration clients
from pycellbase.cbconfig import ConfigClient
from pycellbase.cbclient import CellBaseClient

# Initializing CellBaseClient
cc = ConfigClient("/path/to/config.json")
cbc = CellBaseClient(cc)

# Initializing gene client
gc = cbc.get_gene_client()

# Retrieving transcription factor binding sites (TFBS) for a gene list
gene_list = ['BRCA1', 'BRCA2', 'LDLR']
tfbs_responses = gc.get_tfbs(gene_list, include='id')

# Printing the number of TFBS found for each gene
for response in tfbs_responses:
     print('Number of TFBS for "%s": %d' % (response['id'],
len(response['result'])))
```

Use case

A use case where PyCellBase is used to obtain multiple kinds of data from different sources can be found in this Jupyter Notebook