Programming Interface

Overview

The GREAT Programming Interface allows external clients to submit BED data automatically for analysis by GREAT and display in the GREAT user interface. For example, it is used by the UCSC Table Browser to send data to GREAT. Anyone is welcome to use the interface, and we particularly encourage genome browser portals and ChIP-seq peak calling tools to integrate with GREAT through the GREAT Programming Interface.

Specification

To submit BED data for analysis by GREAT, a client sends an HTTP GET request to `http://bejerano.stanford.edu/great/public/cgi-bin/greatStart.php` that includes:

- `requestURL` (*required*) - The URL for the BED data to process.
- `requestSpecies` (*required*) - The species assembly (e.g. hg18) corresponding to the BED data.
- `requestName` (*optional*) - The name used to identify the BED data on the GREAT output page. If not given, "external data" is used.
- `requestSender` (*optional*) - The name of the tool submitting the data, which is used as a prefix to `requestName` to identify the data on the GREAT output page.
- `bgURL` (*optional*) - The URL for the BED data used as the background for the foreground/background test.
- `bgName` (*optional*) - The name used to identify the background BED data on the GREAT output page. If not given, "external background data" is used.
- `outputType` (*optional*) - The mode in which GREAT returns output.
    1. "`web`" - the standard GREAT UI for presentation in a browser (best for linking from an external website to GREAT)
    2. "`batch`" - a tab separated file for machine processing (best for script-powered analyses)

Submitting an HTTP GET request to GREAT is as simple as including the necessary parameters in a URL link (see examples).

Details

`requestURL`

When GREAT receives a request, it in turn retrieves the BED data at the `requestURL` via HTTP GET. The `requestURL` must point to either a BED file or a script that generates BED data (see examples).

- The `requestURL` should be HTTP encoded so that it is safe to pass via GET (see examples). To encode a URL, enter it at the URL Decoder/Encoder and choose "Encode."

Submitted BED data

The data submitted to GREAT must be valid BED format as used with the UCSC Genome Browser. Lines that start with "#" are considered comments and are ignored. Compressed data is supported.

Batch mode

Batch mode returns a tab-separated text file with GREAT statistical test results for your input data set.

- The format is self-documented with a header explaining the data in each column. The columns are organized to match those of the GREAT web output (when all columns are shown).
- Batch mode returns up to the top 500 ranked terms per ontology.
- Only terms hit by at least one input region are included.
- The footer provides summary statistics on the ontologies tested, along with the number of tests per ontology, which is used in multiple hypothesis test correction.
- The GREAT server will only accept 5 batch queries at a time for all API users. Please see Example 4 below to craft your queries accordingly.
  Importantly, the results returned by batch queries are not filtered.

Examples

Example 1: Simple BED file - web mode

In the first example, the client has a BED file available on its server (at http://www.clientA.com/data/example1.bed) that it wishes for GREAT to process.

http://www.clientA.com/data/example1.bed:

```
# SRF example data
chr5    60663738        60663788        SRF.1
chr5    137828954       137829004       SRF.2
chr5    137828665       137828715       SRF.3
chr7    5536799         5536849         SRF.4
```

To submit this data to GREAT, the client webpage includes a link that sends an HTTP GET request to GREAT. When the link is clicked, the request is sent to GREAT:

```
<html>
 <body>
  <a
href="http://bejerano.stanford.edu/great/public/cgi-bin/greatStart.php?requestSpecies=
hg18&requestName=Example+Data&requestSender=Client+A&requestURL=http%3A%2F%2Fwww.clien
tA.com%2Fdata%2Fexample1.bed">
  Send to GREAT
  </a>
 </body>
</html>
```

Example 2: Script that generates BED data - web mode

In the second example, the client has a script (at http://www.clientB.com/data/example2.php) that generates BED data for GREAT to process.

http://www.clientB.com/data/example2.php:

```
<?php
if ($_GET["tf"] == "Srf") {
     echo "chr5    60663738        60663788        SRF.1\n";
     echo "chr5    137828954       137829004       SRF.2\n";
     echo "chr5    137828665       137828715       SRF.3\n";
     echo "chr7    5536799         5536849         SRF.4\n";
   }
else if ($_GET["tf"] == "Nrsf") {
     echo "chr1    11017025        11017401        NRSF.1\n";
     echo "chr6    12605475        12606176        NRSF.2\n";
     echo "chr8    12680375        12681376        NRSF.3\n";
     echo "chr4    13416275        13416776        NRSF.4\n";
   }
?>
```

Again, to submit this data to GREAT, the client webpage includes a link that sends an HTTP GET request to GREAT. The `requestURL` in turn includes information for GREAT to submit an HTTP GET request back to the client to obtain the BED data:

```
<html>
 <body>
  <a
href="http://bejerano.stanford.edu/great/public/cgi-bin/greatStart.php?requestSpecies=
hg18&requestURL=http%3A%2F%2Fwww.clientB.com%2Fdata%2Fexample2.php%3Ftf%3DNrsf">
  Send NRSF data to GREAT
  </a>
 </body>
</html>
```

Example 3: Simple BED file - batch mode

In the first example, the client has a BED file available on its server (at http://www.clientA.com/data/example1.bed) that it wishes for GREAT to process. The client wants to do further automated processing, and thus wishes to have the GREAT results in a machine-readable tab-separated format.

http://www.clientA.com/data/example1.bed:

```
# SRF example data
chr5    60663738        60663788        SRF.1
chr5    137828954       137829004       SRF.2
chr5    137828665       137828715       SRF.3
chr7    5536799         5536849         SRF.4
```

To submit this data to GREAT, the client issues a "wget" request and specifies "`outputType=batch`":

```
wget -O results.tsv
"http://bejerano.stanford.edu/great/public/cgi-bin/greatStart.php?outputTy
pe=batch&requestSpecies=hg18&requestName=Example+Data&requestSender=Client
+A&requestURL=http%3A%2F%2Fwww.clientA.com%2Fdata%2Fexample1.bed"
```

Example 4: Running many jobs in parallel - batch mode

The GREAT server currently accepts a maximum of 5 API requests for all API users at any given time. More than 5 requests will be denied. To make the maximum of 5 API calls
you will need to split your jobs and process them separately. A possibly way of processing the data is seen in our greatBatchQuery.py script. Feel free to adapt it to your needs.

How does greatBatchQuery.py work?

- Create a joblist of the format: `outputFile queryURL` where each line is a new job
  - EX:

```
results1.tsv
http://bejerano.stanford.edu/great/public/cgi-bin/greatStart.php
?outputType=batch&requestSpecies=hg18&requestName=Example+Data&r
equestSender=Client+A&requestURL=http%3A%2F%2Fwww.clientA.com%2F
data%2Fexample1.bed
results2.tsv
http://bejerano.stanford.edu/great/public/cgi-bin/greatStart.php
?outputType=batch&requestSpecies=hg18&requestName=Example+Data&r
equestSender=Client+A&requestURL=http%3A%2F%2Fwww.clientA.com%2F
data%2Fexample2.bed
```

- Then run `./greatBatchQuery.py joblist`