Galaxy External Tool Adding Tutorial

1. First, Galaxy requires Python 2.7. To check your python version, run:

   ```
   python –V
   ```

   The output should be:

   ```
   Python 2.7.3
   ```

2. Then, please clone the Galaxy Project GitHub repository:

   ```
   git clone https://github.com/galaxyproject/galaxy.git
   ```

   Start Galaxy by running the command:

   ```
   sh run.sh
   ```

   Once Galaxy completes startup, you should be able to view Galaxy in your browser at:

   http://localhost:8080

3. After you see that it runs fine, create an account and login to Galaxy. After you've logged in, you must add that account as an Administrator to do further modifications in order to add a tool.
   a. Add the Galaxy login (email) to the Galaxy configuration file config/galaxy.yml.
   b. In the first-time installation, the file does not exist.
   c. If the file does not exist yet, you can create it from the provided sample (config/galaxy.yml.sample) (copy as config/galaxy.yml)
   d. Make sure you rename it correctly (config/galaxy.yml)
   e. NOTE: you must restart Galaxy after modifying ANY configuration file for changes to take effect.
   f. Also, do not specify the email line twice - the line with admin_users are already present (at line 1361 if you copy galaxy.yml.sample file as it is, so modify the null to your email address) in every sample config and should be there only once.

      ```
      # this should be a comma-separated list of valid Galaxy users
      admin_users: user1@example.com,user2@example.com
      ```

4. Restart Galaxy. If your email address is successfully added to the admin_users list, you will see an "Admin" menu item in the top Galaxy menu bar which will take you to the Galaxy Admin page.

5. After you see the Admin menu item in the top Galaxy menu bar, it is time to proceed to adding a tool to Galaxy. For this tutorial, we will add Racon tool to our local Galaxy instance.

6. Before we add Racon to the instance, let us test it outside the Galaxy instance to check if we have the dependencies. We need the CUDA support for the tool version we want to add, so the dependencies are:

```
gcc 4.8+ or clang 3.4+
cmake 3.2+

CUDA Support
gcc 5.0+
cmake 3.10+
CUDA 9.0+
```

7. Go to the “tools” directory of Galaxy:

```
cd tools
```

8. Get the Racon repository to tools directory:

```
git clone --recursive https://github.com/lbcb-sci/racon.git racon
cd racon
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
```

or for GPU supported build, type in the command:

```
cmake -DCMAKE_BUILD_TYPE=Release -Dracon_enable_cuda=ON ..
```

then, type in:

```
make
```

The above command will create an executable in the bin file, which is inside the build (current) folder.

9. If there are no errors in the previous step, proceed to run the Racon tool with the below commands:

```
cd bin
./racon                              ../../test/data/sample_reads.fastq.gz
../../test/data/sample_overlaps.paf.gz
../../test/data/sample_layout.fasta.gz
```

Detailed Documentation of Racon:

```
racon [options ...] <sequences> <overlaps> <target sequences>
    <sequences>
    input file in FASTA/FASTQ format (can be compressed with gzip)
    containing sequences used for correction
    <overlaps>
    input file in MHAP/PAF/SAM format (can be compressed with gzip)
    containing overlaps between sequences and target sequences
    <target sequences>
    input file in FASTA/FASTQ format (can be compressed with gzip)
    containing sequences which will be corrected
```

```
 options:
    -u, --include-unpolished
    output unpolished target sequences
    -f, --fragment-correction
   perform fragment correction instead of contig polishing (overlaps
   file should contain dual/self-overlaps!)
    -w, --window-length <int>
    default: 500
    size of window on which POA is performed
    -q, --quality-threshold <float>
    default: 10.0
    threshold for average base quality of windows used in POA
    -e, --error-threshold <float>
    default: 0.3
    maximum allowed error rate used for filtering overlaps
    --no-trimming
    disables consensus trimming at window ends
    -m, --match <int>
    default: 3
    score for matching bases
    -x, --mismatch <int>
    default: -5
    score for mismatching bases
    -g, --gap <int>
    default: -4
    gap penalty (must be negative)
    -t, --threads <int>
    default: 1
    number of threads
    --version
    prints the version number
    -h, --help
    prints the usage

 only available when built with CUDA:
    -c, --cudapoa-batches
    default: 1
    number of batches for CUDA accelerated polishing
    -b, --cuda-banded-alignment
  use banding approximation for polishing on GPU. Use if -c is used.
    --cudaaligner-batches
    Number of batches for CUDA accelerated alignment
```

If this runs successfully, let's add it to Galaxy instance.

10. Make sure you are inside the racon directory. Create two files called racon.xml and macros.xml and copy the contents of https://raw.githubusercontent.com/bgruening/galaxytools/master/tools/racon/racon.xml to racon.xml and contents of https://raw.githubusercontent.com/bgruening/galaxytools/master/tools/racon/macros.xml to macros.xml. Then, open racon.xml and edit line 17 and replace "racon" and give the absolute path of the executable that was created in step 8. It should look like below:

```
    <tool id="racon" name="Racon" version="1.3.1.1">
      <description>Consensus module for raw de novo DNA assembly of long
uncorrected reads.</description>
      <macros>
      <import>macros.xml</import>
      </macros>
      <expand macro="requirements" />
      <version_command>racon --version</version_command>
      <command detect_errors="exit_code"><![CDATA[
      ln -s '$reads' reads.${reads.ext} &&
      #if $overlaps.ext == 'sam':
          ln -s '$overlaps' overlaps.${overlaps.ext} &&
      #else:
          ln -s '$overlaps' overlaps.paf &&
      #end if
    ln -s '$corrected_reads' corrected_reads.${corrected_reads.ext} &&
    /home/$USER/galaxy/tools/racon/bin/racon
  reads.${reads.ext}
...
```

11. We now must make Galaxy aware of the Racon tool. To do this, we will add a section to the tool_conf.xml file, which is in the config/ directory.

```
    <section name="racon" id="racon">
      <tool file="racon/racon.xml" />
    </section>
```

12. After you did step 11, restart Galaxy and in http://localhost:8080/ and make sure the Admin tab is there. Then, click the Analyze Data tab. On the left, click Get Data, and then "Upload File from your computer" from the dropdown menu, an upload bin will pop-up. Then, from the file explorer, open /home/$USER/galaxy/tools/racon/test/data directory and drag and drop all the .gz files to the upload bin in Galaxy.

13. These are the input files that we must give to Racon to test its functionality using Galaxy. Now, in the Galaxy tool, on the left pane, there should be a search box for searching the tools. Search the name of the tool you added.

14. For this tutorial, the name of the tool was Racon, so search Racon. It should be available in the search results. Click Racon and after it opens, you should see 3 input data boxes and a bunch of parameters specifying boxes. For the first input data, you must select sample_reads.fastq.gz for the second, sample_overlaps.paf.gz and for the third, sample_layout.fasta.gz to test. Then, scroll down and click "Execute". The job will be added to the queue and it will be completed shortly. When it is completed, you will see that it will turn green on the right History pane. You can see the details. If there is an error, it will turn red, so you must check the details for the source of error.

If you have any questions, email me (Gulsum Gudukbay, gulsum@psu.edu).

Resources:

1. https://github.com/galaxyproject/galaxy
2. https://github.com/lbcb-sci/racon
3. https://github.com/bgruening/galaxytools/tree/master/tools/racon
4. https://galaxyproject.org/admin/
5. https://galaxyproject.org/admin/tools/add-tool-tutorial/
6. http://wiki.sb-roscoff.fr/ifb/index.php/Tool_Integration_Short_Tutorial
7. https://galaxyproject.org/admin/config/tool-dependencies/