PeakLearner: User Manual

GNomes

Members
Jacob Christiansen
Allen Clarke
Yuanyuan Fu
John Jackson

Mentor
Mahsa Keshavarz

Clients
Dr. Toby Hocking
Christopher Coffey

Version 1.0
May 7, 2020

Contents

1 Introduction

Thank you for choosing PeakLearner for your genomic data analysis. PeakLearner is a genomic data viewing tool that has been custom-designed to meet your needs. Some of the key highlights include:

- label drawing with color-coded labels
- dynamic model redrawing based on supervised machine learning
- the same look and feel as other genome browsers

The purpose of this user manual is to help you, the client, successfully install, administer, and maintain the PeakLearner product into your existing system and to make sure it is as useful as possible for years to come.

2 Installation

As part of final delivery, the PeakLearner system is capable of running on any Ubuntu 16.04 or Mac 10.15.04 (Catalina) machine. Over time, however, you may want to move to a new platform or re-install the product. On an Ubuntu machine, use the following commands to install NodeJS, Berkeley-DB version 6.1, and the Python package bsddb3:

- `apt-get install nodejs`
- `brew install berkeley-db`
- `apt-get install python3-bsddb3`

To install on OSX Catalina, use these commands:

- `brew install node`
  - If you encounter this error: `Warning: the post-install step did not complete successfully`
    run the following commands:
    - `sudo chown -R <username> /usr/local/lib/node_modules/npm/node_modules`
    - `sudo chown -R <username> /usr/local/lib/node_modules/npm/`
    - `brew postinstall node`

- `brew install berkeley-db`
- `pip install bsddb3`

Once all of these packages are installed properly, you will have everything needed to set up the server used to communicate with the web browser.

The next step is going to the PeakLearner github repository `https://github.com/yf6666/PeakLearner-1.1` and cloning the master branch, as it is the most stable branch. Once this is done, you will have to configure JBrowse to work properly by following this sequence of steps.

1. `git clone https://github.com/yf6666/PeakLearner-1.1`
2. create a new branch if desired
3. from the /jbrowse directory, run `./setup.sh`
4. make a new directory called `data`

For additional information on setup and installation, read the README files in the jbrowse folder and the outermost directory.

3 Configuration and Daily Operation

JBrowse requires a server, and you can run the server by simply typing `./ourServer.py.`

This will start a web server hosting PeakLearner on the desired port found in the ourServer.py file. To have the ability to use an InteractivePeakAnnotator track you will also need to start the restAPI.py by typing `python restAPI.py.`

This will start a side server on port 5000. Then you can go to `http://localhost:<port#>/?data=plugins/InteractivePeakAnnotator/test/data` for a small demo with the volvox data that comes with JBrowse.

When you want to show your own data you will need to create a tracks.conf file. You will most likely place this file inside of the data directory created in the setup. Once this is done, `http://localhost:<port#>` will immediately open that directory and the tracks.conf within. If you have many directories you can place a tracks.conf in each and switch between them by using data=<folder path> inside of your url. For example: http://localhost:<port #>/?data=plugins/InteractivePeakAnnotator/test/data

Below is an example tracks.conf with a breakdown of some of its terms:

```
[GENERAL]
refSeqs=volvox.fa.fai

[tracks.refseq]
urlTemplate=volvox.fa
storeClass=JBrowse/Store/SeqFeature/IndexedFasta
type=Sequence
key=volvox reference

[tracks.my-bigwig-track]
storeClass = JBrowse/Store/SeqFeature/BigWig
urlTemplate = myfile.bw
type = JBrowse/View/Track/Wiggle/XYPlot
key = Coverage plot of NGS alignments from XYZ
```

- **General**: a heading that denotes where information that will apply to all the tracks in the file. The only required assignment here is "refSeqs" as this is used by JBrowse to know where in the file the user is currently looking across all of the tracks in the file.
- **tracks.<NAME>**: each track needs to have a unique tracks.<NAME> to denote it. This name is passed to the server when adding and removing labels. Below this will be all of the setup for the track in question.
- **urlTemplate**: the relative path or url to the file
- **storeClass**: this dictates how the file is stored by jbrowse. there are more types of store classes then what is shown above, to find some of the other ones for basic file types you can go to "jbrowse/src/Jbrowse/store/SeqFeature". The other important kinds of storeClass is those from plugins. To find these go in to "jbrowse/plugins/<PLUGIN NAME>/js/Store/SeqFeature".
- **type**: designates the track type, and determines how the file responds to user input. Some basic types are "Sequence" and "Alignments2". For some more specific ones you can go to "JBrowse/ View/Track/<TRACK TYPE>/". Just as with storeClass type can also be inside of a plugin at "jbrowse/plugins/<PLUGIN NAME>/js/ View/Track"
- **key**: the name that will actually be displayed in the browser

Below is an example of an InteractivePeakAnnotator track:

```
[tracks.interactive]
key=Interactive MultiXYPlot
type=InteractivePeakAnnotator/View/Track/MultiXYPlot
urlTemplates+=json:{"url":"coverage.bigWig",
"name":"volvox_positive", "color":"#235"}
urlTemplates+=json:{"storeClass":"JBrowse/Store/
SeqFeature/REST", "baseUrl":"http://127.0.0.1:5000",
"name":"joint_peaks", "color":"red", "lineWidth":5,
"noCache":true}
storeClass=MultiBigWig/Store/SeqFeature/MultiBigWig
storeConf=json:{"storeClass":
"InteractivePeakAnnotator/Store/SeqFeature/Features"}
```

This is much like a normal bigWig track but with a few differences, the first being urlTemplates. Instead of a path or url you will add new JSON objects with some information on the bigWig. It is important to notice it is '+=' for urlTemplates not an '='. These JSON objects can be thought of as mini track configurations for the bigWig inside.

For the second urlTemplate you define a different storeClass, the this one being for a REST file. This, along with noCache being true, is what makes it the model update immediately. The REST store class makes it so that JBrowse asks an outside API in order to get the sequence information and the noCache means that JBrowse wont store the information locally, so once the model updates on the server side it will update on the user side.

4 Maintenance

The source code can be found here to reinstall/setup the system: `https://github.com/yf6666/PeakLearner-1.1`

Separately, the majority of the action for the server is found in the ourServer.py file. The send_post is what handles adding labels to the database and is also where the code skeleton to hook up the system to a GPU cluster can be found. The majority of the server should be very stable, but any server issues can be handled in this file.

The code we have added to JBrowse can primarily be found inside of the main.js and MultiXYPlot.js files inside of InteractivePeakAnnotator. The links to the repositories that this code releases on can be found in the README inside of the JBrowse folder of PeakLearner.

Inside of main.js you will find the event listener to watch the highlight tool of jbrowse to create new labels. MultiXYPlot.js has the code to edit and remove labels.

The only other thing JBrowse will rely on would be the restAPI.py file. Note this is only used by tracks with the rest store class, including the setup discussed above for interactivePeakAnnotator tracks. For more info about the JBrowse REST API you can look here: `https://jbrowse.org/docs/data_formats.html`.

5 Troubleshooting

The best way to troubleshoot the system is to add logs to ensure the system is communicating as intended. The most common errors happen when data being sent between two entities is not properly encoded/decoded and is sent to the wrong location. The easiest way to fix this is to print to the console exactly what the server is sending and receiving whenever the browser makes a request. This will eliminate the majority of communication problems and make sure that the data is in the corrected format.

To troubleshoot any errors pertaining to JBrowse and the model displaying, you should first open the console on your browser. JBrowse will print out many errors and why they happened there. If that does not help, try the following:

- from /PeakLearner-1.1/jbrowse, run `bin/generate-name.pl`
- uninstall and reinstall JBrowse and all of its dependencies

The latter step can be accomplished by deleting the local save of PeakLearner and again going through the setup process.

6 Conclusion

This concludes the PeakLearner user manual. We wish you years of productive use of this product, and we are happy to have been of service.

While we are all moving on to professional careers, we would be happy to answer questions in the coming months to help this project's continued development and operation. With best wishes from your PeakLearner developers:

- Jacob Christiansen (jdc465@nau.edu)
- Allen Clarke (amc879@nau.edu)
- Yuanyuan Fu (yf66@nau.edu)
- John Jackson (jcj52@nau.edu)