SCHMITZ'S LAB JBROWSE REFERENCE V3

BRIGITTE HOFMEISTER

Contents

1. Important Information

1.1. Server.

- URL: epigenome.genetics.uga.edu
- Username: `schmitzlab1`
- Password: `schmacct5$`
- Path to main JBrowse: /Library/WebServer/epigenome/JBrowse/
- To access this server via terminal, you must be on the UGA campus or use VPN.
- Anything added to server gets very restricted permissions by default. After adding anything, run `chmodd` and `chmodf` (these are aliases to custom functions that recursively change all directories to `755` and files to `644`)

1.2. ZCluster.

- Path to main JBrowse: /home/rjslab/JBrowse/
- Path to custom python scripts: /home/rjslab/JBrowse/scripts/

1.3. Memory Management Guidelines.

- To avoid overloading the server and minimize the amount of data transferred, most of the work should be done on the zcluster (sapelo eventually) or personal computer.
- JBrowse is easy to install, so all the work can be done on a personal computer then transferred to the server.
- Creating a species and file conversions should be done on escratch or personal computer.
- To not overload the home directory, converted files can be directly uploaded to the server.
- Editing the tracks CSV file can be done anywhere, but I find using Excel the easiest
- The generating-names step can be done on the zcluster or server; it needs to be run with all of the annotation/peak tracks within the directory. After initially creating the species, I'd suggest running it on the zcluster before uploading. For additional tracks, run on the server.
- I strongly suggest using `rsync` for uploading to the server, especially for the raw data files.
  a) `rsync` only updates the files that have changed
  b) `rsync` has a "partial" option where it will continue uploading where it left off if it was previously interrupted.
  c) For initial species creation: `rsync -az` *zcluster_folder* `schmitzlab1@epigenome.genetics.uga.edu:../JBrowse/`
  d) For large files: `rsync -az -P` *zcluster_folder* `schmitzlab1@epigenome.genetics.uga.edu:../JBrowse/`*path_to_folder*

1.4. Github.

- Username: `schmitzlab`
- Password: `schm@cct8.`
- Lab browser meta-data repository: `https://github.com/schmitzlab/jbsite-lab.git`

- Plant methylome meta-data repository: `https://github.com/schmitzlab/jbsite-plantmethyl.git`
- Python scripts and browser plugins: `https://github.com/bhofmeis`
- To keep everything up-to-date across multiple computers, the meta-data files for the species within JBrowse are maintained in a github repository. This repository does not store any raw data files, reference sequence, or annotations. It only includes configuration files and track files for the main website and each species.
- If you want to do work on your personal computer, use

  `git clone https://github.com/schmitzlab/jbsite-lab.git`

  to get the repository.
- The repository already exists in the JBrowse folder on zcluster and on the server's
- When you have created a new species, transfer the data to the server. On the server in the main JBrowse folder, run the following commands:

  `git pull` `git add` *species_folder*
  `git add jbrowse.conf`
  `git commit -m` *message indicating what you changed*
  `git push`

  This updates the existing files, adds the new species files, the pushes the new information to github account. It will likely ask for a username and password; use the one for the github account.
- When adding new data tracks to an existing species, transfer the raw data files to the server. On your personal computer, the JBrowse folder on zcluster or server, run

  `git pull`

  to get current versions of meta-data files.
  Then edit the track listing file appropriately. To push these changes to the github account, run

  `git add` *species_folder*
  `git commit -m` *message indicating what you changed*
  `git push`

  If the following was done on a personal computer or the zcluster, in the main JBrowse folder on the server, run

  `git pull`

2. Creating a New Species

2.1. **Preparing the Genome.**

a) Determine a Dataset Name

We need data for a single species together but separate from other species. This is accomplished with the file hierachical structure. Each species gets a folder.

Choose a datasset name for this new species. If the species has a popular common name, use this, i.e. "humans" for *Homo sapeins* or "arabidopsis" for *Arabidopsis*

thaliana. If there isn't a common name (or the common name isn't common), the dataset name will be in the following format: "hsapeins" or "a_thaliana".

This dataset name needs to be unique among all speices in the browser, so check that it isn't already used. For the rest of the documentation, we will refer to this chosen dataset name as the *commonname*.

b) Creating Species Folder

On your personal computer or escratch of the zcluster, create a folder called commonname. In the commonname folder, add a folder called raw that will store the raw data. Within the raw folder, add folders chip, atac, methyl, and rna for ChIP-seq, ATAC-seq, methylation, and RNA-seq data, as applicable.

Preferably, create a folder within the raw folder called annotation. This folder will store compressed versions of the FASTA and GFFs. The files in this folder won't be used by the browser, however, but will make it easy to rebuild the reference sequence and annotation tracks if the JSON-formatted versions are lost or corrupted.

c) Trimming and Formatting the FASTA

To make trimming and formatting much easier, there is a python script prepare_fasta_for_broswer.py to help. It's main aim is to rename chromosomes/scaffolds/contigs consistently across the browser. It also reorders the sequences to be in numerical order. Additionally, it can filter out chromosomes that we don't want to include, i.e. scaffolds or lambda DNA. Scaffolds should be eliminated from the FASTA when annotated chromosomes contain the vast majority of the annotated genome. You can also filter out chloroplast, mitochondria, and lambda DNA. Finally, you can specify to only include a set list of chromosomes. This list needs to be comma-separated and match identically to the sequence names in the file.

Once the FASTA has been trimmed/formatted by the script, run `samtools faidx` to (a) verify the correct sequences are included and (b) create file with chromosome sizes needed later.

Using prepare_fasta_for_broswer.py:

```
python3 prepare_fasta_for_browser.py [-no-scaf] [-no-clm] [-i=chrm_list]
<fasta_file>
-no-scaf    does not include scaffolds/contigs in the output
-no-clm    do not inlclude chroloplast, mitochondria, and lambda in output
-i=chrm_list    when specified, only includes these chromsomes in output;
comma-separated list
```

d) Uploading to JBrowse

Using the trimmed and formatted FASTA and run one directory above the commanname folder:

```
bin/prepare-refseqs.pl --fasta <fasta_file> [--out commonname]
```

This will create a seq folder within the commonname folder.

2.2. **Preparing the Annotation.**

This step is fairly cumbersome, but the work put into this now, the better it will be in the browser.

a) Filtering and Formatting the GFF File

If your species has repeats annotated in a separate GFF file, combine the GFF files into one; this makes formatting easier.

There are two types of filtering for GFF files, both accomplished with one script: prepare_gff_for_browser.py.

First, some species have non-coding RNAs (snoRNA, ncRNA, tRNA, rRNA, miRNA, ect) annotated within the main GFF file. We want these in a separate GFF for a separate "Non-Coding RNA" track. Similarly, the transposable elements or repeats listed within the GFF need to be separated.

Second, JBrowse annotation tracks function best when only including the "gene", "mRNA", "five_prime_UTR", "CDS", and "three_prime_UTR" feature types. It acts weird with "exon", "intron", "protein", and 'chromosome". So we filter this out. If your GFF does not have "CDS", you need to convert "exon" to "CDS".

At the same time as filtering, we want to make sure the chromosome/scaffold names match the formatted FASTA names. The program does this reformatting by default, but can be turned off. Similar to formatting the FASTA file, you can specify to remove annotation for scaffolds and chloroplast, mitochondria, and lambda DNA. This is not necessary for the browser, but helps with memory space.

Using prepare_gff_for_broswer.py:

```
python3 prepare_gff_for_browser.py [-no-clean] [-rna] [-tes] [-rpt] [-o=output_prefix]
[-no-scaf] [-no-clm] <gff_file>
-no-clean    do not rename chromosomes
-rna    GFF has ncRNA which should be separated out
-tes    GFF has transposons which should be separated out
-rpt    GFF has repeats, annotated as ‘‘similarity’’, which should be
separated out
-no-scaf    does not include scaffolds/contigs in the output
-no-clm    do not inlclude chroloplast, mitochondria, and lambda in output
-o=output_prefix    an optional output file prefix name
```

b) Adding Ortholog Information

One of the coolest features of this JBrowse is the ability to jump to orthologous genes in other species. This is accomplished by including that information in the GFF file. You need a file that contains the names of orthologous genes in two species with each gene on one line. The script add_ortholog_gff.py will produce new GFF files for both species.

Using add_ortholog_gff.py:

```
python3 add_ortholog_gff.py [-l=label1,label2] [-c=col1,col2] <ortholog_file>
<species1_gff> <species2_gff>
ortholog_file    file of orthogous genes; genes listed first correspond
```

```
  to 'species 1' and those second are for 'sepecies 2'
species_gff    corresponding GFF files for the species; order is important
  and should match the ortholog file
-l=label1,label2    label information when encoding the ortholog in the
  GFF; ideally, use this style:  Hsapiens, Athaliana, Esalsugineum
-c=col1,col2    0-based index of the column that includes the gene names
  for species 1 and 2; [default 0,1 (only gene names listed per line)]
```

c) Uploading to JBrowse

There are separate uploads for each GFF file (genes, rna, transposons, repeats). One directory above the commanname folder:

`bin/flatfile-to-json.pl --gff <`*gff_file*`> --trackLabel <`*label*`> [--out`*commonname*`]`

Please use exactly "genes", "rnas", "te", "repeats" for the *label*, as appropriate

2.3. Assigning as a Dataset.

a) Updating jbrowse.conf

In the main JBrowse folder, run

`git pull`

Open jbrowse.conf folder. In the middle of the file you will see a list of datasets. Under the existing data sets, add the lines:
`[datasets.`*commonname*`]`
`url = ?data=`*commonname*
`name =` *Species name*
Be sure to push these changes to github using `git add`, `git commit`, then `git push`.

b) Updating tracks.conf

Within the species folder commonname, open the tracks.conf file. We need to connect this species to the datasets. This file should be empty, so add:
`[general]`
`dataset_id =` *commonname*

Also, if the species genome consists of mainly scaffolds, add this line:
`refSeqOrder = length descending`

2.4. Making Gene Names Searchable.

To be able to search by gene name in the browser, JBrowse needs to index and store that information in it's own format.

One directory above the commanname folder, run:

`bin/generate-names.pl -v --out` *commonname*

This will print out information about the tracks and some numbers. Eventually there will be a line like `Using 1 chars for sort log names (16 sort logs)`. If this line says "Using 2 chars", immediately stop the program and re-run with this additional option:

```
--mem 5000000000
```

There's a bug somewhere in their code and this is the easiest way around it. Increase `mem` as necessary until only one `chars` is used.

This step will likely take awhile depending on number of genes and genome size.

2.5. **Setting Up The Track List.**

In the JBrowse docs folder there is a tracks_empty.csv file. Copy this into the new species folder. In the species folder, change the name of this file to be more informative, i.e. tracks_commonname.csv.

JBrowse knows about the data tracks from the trackList.json file, but this file is cumbersome to edit directly. To make adding tracks easier and have the formatting of tracks be consistent, there is a script that will convert the information from the CSV to the JSON format. More detailed information is below.

3. Converting Files

`bedtools`, `bedGraphToBigWig`, and `bedSort` should be on the path.

On the zcluster, `bedGraphToBigWig` and `bedSort` are stored in the JBrowse/scripts folder. So simply add `export PATH=$PATH:/home/rjslab/JBrowse/scripts` to your .bashrc file.

3.1. **allC Files.**

There is one super-handy script that will convert allC files to a special BigWig format. Unlike the methylpy output, we want the allC information for all of the chromosomes in one file. The script formats chromosome names to match the formatting produced by prepare_fasta_for_broswer.py and prepare_gff_for_browser.py and only includes chromosomes listed within the chromosome sizes file.

For a single species, the allc_to_bigwig_pe.py script can take in numerous allC files, which makes conversion even easier. The script also needs a file with chromosome sizes; I'd suggest using the fasta index (.fa.fai) for the formatted genome.

Using allc_to_bigwig_pe.py:

```
python3 allc_to_bigwig_pe.py [-keep] [-no-clean] [-l=labels] [-p=num_proc] [-o=out_id]
<chrm_sizes> <allC_file> [allC_file]*
allC_file    allC format file with all the chromosomes
chrm_sizes    tab-delimited file with chromosome name and size
-keep    when on, keeps the intermediate files
-no-clean    does not rename chrms or check chrm names match chrm file
-l=labels    comma-separated list of labels to use for the allC files; defaults
to using information from the allc file name
-o=out_id    optional identifier to be added to the output file names
-p=num_proc    number of processors/threads to use [default 1]
```

3.2. **Sequencing BED files and BAM files.**

For ATAC-seq, ChIP-seq and RNA-seq, we want a coverage view. There is one script for this conversion and it is similar to the allC file conversion. You can input both BED

files and BAM files and it will convert those to BigWig format. The script formats chromosome names to match the formatting produced by prepare_fasta_for_broswer.py and prepare_gff_for_browser.py and only includes chromosomes listed within the chromosome sizes file. There are a variety of options, but in general you only want to use the scaling option.

Using file_to_bigwig_pe.py:

```
python3 file_to_bigwig_pe.py [-keep] [-no-clean] [-strand] [-scale] [-union]
[-p=num_proc] <chrm_file> <bam_file | bed_file> [bam_file | bed_file]*
chrm_file    tab-delimited file with chromosome names and lengths
bam_file    bam file that already has been indexed
bed_file    BED formatted file
-keep    keep intermediate files
-strand    separate reads by strand to have strand-specific bigwig files
-scale    scale the bigwig values by total number of reads in file
-union    used with strand, combine stand-specific values into one bigwig file
-no-clean    does not rename chrms or check chrm names match chrm file
-p=num_proc    number of processors to use [default 1]
```

3.3. **Peak/Annotation BED files.**

Peak finders usually give results as BED files, which is fine for directly uploading.

They need to be converted to GFF if you want to locally upload the file via the webpage. Using the webpage is useful for examining peaks/testing out peak finders. There is one simple python script for this, bed_to_gff.py.

Using bed_to_gff.py: `python3 bed_to_gff.py [-i=source] <bed_file>`

```
bed_file    BED file to convert
-i=source    string indicating the source of the BED file
```

4. Adding and Organizing Tracks

4.1. **Including the Raw Data.**

a) Peak Finding

Peaks should be in BED or GFF format. From the main JBrowse directory, they are uploaded using:

BED: `bin/flatfile-to-json.pl --bed <peak_file> --trackLabel label`
GFF: `bin/flatfile-to-json.pl --gff <peak_file> --trackLabel label`

Make note of the name used for the label

b) Methylation

Upload the converted BigWig file into the species raw/methyl/ folder. Make sure file names are unique as not to overwrite existing data.

c) ChIP-seq

Upload the converted BigWig file into the species raw/chip/ folder. Make sure file names are unique as not to overwrite existing data.

d) ATAC-seq

Upload the converted BigWig file into the species raw/atac/ folder. Make sure file names are unique as not to overwrite existing data.

e) RNA-seq

Upload the BigWig file into the species raw/rna/ folder. Also upload the corresponding BAM file and BAM index (.bam.bai) file into this same folder.

4.2. **Adding the Track.**

The track information will be stored in the tracks_species.csv file. If updating this file on a personal computer, please download the most up-to-date version to before editing. I find Excel the easiest way to edit this.

Each line represents a separate track. Lines that begin with `#` are considered comments and are ignored.

The file tracks_empty.csv contains examples, as comments, for each of the track types. Uncomment the lines needed, particularly for `DNA` and `genes`. The easiest way to add a new track is copy-paste-update of the appropriate example.

Columns are defined below.

i. trackType: possible options are `ataseq`, `DNA`, `genes`, `rnas`, `te`, `repeats`, `rnaseq`, `methyl`, `chip`, `peaks`, `reads`
ii. label: unique label for the track; for tracks added via GFF file, the label listed here must match the label used in the original upload
iii. key: this is the visible label for the track in the browser
iv. category: category to group this track; use "/" to indicate a subcategory, i.e. if B is a subcategory of A, use "A/B"
v. chip type/orthologs/color/height
    1. If a common histone modification (chip track and peaks track), indicate the modification here; use lowercase and "m" for methylation, i.e. H3K4me3 → "h3k4m3"
    2. For other ChIPs, like TF, indicate the color you want to use
    3. For the `genes` track, this is a semi-colon separated list of the *commonname* for orthologs included in the annotation, i.e. *A. thaliana* is linked to *P. trichocarpa* and *E. salsugium*, so use "poplar;eutrema"
    4. For RNA-seq, this is a height parameter for the maximum score when displaying histograms. I'd suggest using the 95% percentile of values from the BigWig file. Can specify minimum and maximum as `min;max`
    5. For generic reads type, i.e. the reads for ATAC-seq or Chip-seq, this is folder name within raw where the BAM is stored.
vi. Chip/methyl/rna bigwig file: name of the BigWig file for this track; name of the file only, the script will add the appropriate folder information
vii. Rna-seq bam: name of the BAM file; this is the name of the BAM file only and the BAM file must have the associated index file
viii. Genome version/description

1. For `DNA`/annotation: string indicating the genome version
2. For all other tracks, include a brief description for the track

ix. Source label/GGF run
   1. For `DNA`/annotation, a string/label for the web address to the genome, i.e. "arabidopsis.org"
   2. For published/SRA data, indicate SRX/SRR number, i.e. "SRX551765"
   3. If from GGF, include the run number, i.e."run 6"

x. Source link: for DNA/annotation/published data, include the URL to the source

xi. Mapping rate: include the overall read mapping rate, if known

xii. Percent remaining: include the percent remaining, number mapped reads / number raw input reads, if known

xiii. Metadata: additional metadata for the track; needs to formatted as `Key:Value` separated by semi-colons

4.3. **Converting the track list.**

The tracks_species.csv file is converted with the build_tracklist_json.py program. As it converts the file, it prints the label for each track. Double check that all tracks are included and look out for any error messages. This script automatically creates the trackList.json file in the same directory as the tracks_species.csv file, overwriting the existing trackList.json file.

Also, if necessary, add the appropriate ortholog formatted name and *commonname* name to the dictionary towards the bottom of the build_tracklist_json.py script.

Using build_tracklist_json.py on Zcluster:

`python /home/rjslab/JBrowse/scripts/build_tracklist_json.py <`*track_info_file*`>`

Using build_tracklist_json.py on Server:

`python` *JBrowse*`/bin/build_tracklist_json.py <`*track_info_file*`>`

5. Quick Guide

1. Creating a New Species
   a. Choose the *commonname*
   b. Make folders commonname and commonname/raw
   c. `python3 prepare_fasta_for_browser.py [-no-scaf] [-no-clm] [-i=chrm_list] <`*fasta_file*`>`
   d. `samtools faidx <`*formatted_fasta_file*`>`
   e. `prepare-refseqs.pl --fasta` *fasta_file* `--out` *commonname*
   f. Combine GFFs if necessary
   g. `python3 prepare_gff_for_browser.py [-rna] [-tes] [-rpt] [-o=output_prefix] [-no-scaf] [-no-clm] <`*gff_file*`>`
   h. `python3 add_ortholog_gff.py [-l=label1,label2] [-c=col1,col2] <`*ortholog_file*`> <`*species1_gff*`> <`*species2_gff*`>`
   i. `flatfile-to-json.pl --gff`*gff_file*`--trackLabel` *label* `--out` *commonname*

j. `generate-names.pl -v --out` *commonname*
k. Optionally, add gzipped FASTA and GFFs to commonname/raw/annotation
l. Copy tracks_empty.csv file to commonname and edit tracks.json
m. Copy commonname folder to server with `rsync -az`
n. From main JBrowse directory on the server , update git repository:
`git add` *commonname/*
`git add jbrowse.conf`
`git commit -m` *message*
`git push`
o. From species fold on the cluster change permissions:
`chmodd chmodf`

2. Converting Files
a. `python3 allc_to_bigwig_pe.py [-keep] [-no-clean] [-l=labels] [-p=num_proc] [-o=out_id]` <*chrm_sizes*> <*allC_file*> [*allC_file*]*
b. `python3 file_to_bigwig_pe.py [-keep] [-no-clean] [-strand] [-scale] [-union] [-p=num_proc]` <*chrm_file*> <*bam_file* | *bed_file*> [*bam_file* | *bed_file*]*
c. `python3 bed_to_gff.py [-i=source]` <*bed_file*>

3. Adding and Organizing Tracks
a. Peak Files: `flatfile-to-json.pl --bed` <*peak_file*> `--trackLabel` *label* `--out` *commonname*
b. allC Files: BigWig into raw/methyl/
c. ChIP-seq Files: BigWig into raw/chip/
d. RNA-seq Files: BigWig, BAM, and BAM index into raw/rna/
e. Transfer raw folder to server with `rsync -az -P`
f. Run `git pull` or `git clone https://github.com/schmitzlab/jbsite-lab.git` to get the most up-to-date species track listing
g. Update tracks_species.csv
h. `python build_tracklist_json.py` <*track_info_file*>
i. From location with updated track list, pdate git repository:
`git add` commonname
`git add jbrowse.conf`
`git commit -m` *message*
`git push`
j. On the server,
`git pull` in main JBrowse folder
`chmod` in commonname folder
`chmodf` in commonname folder