WashU Epigenome Browser Documentation

Release 1.0

Daofeng Li, Silas Hsu, Renee L. Sears and Ting Wang

Nov 26, 2018

Contents:

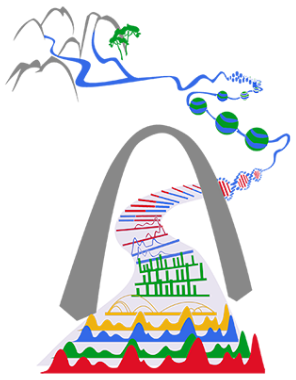
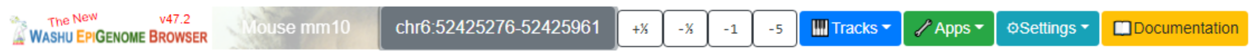


Fig. 1: Gateway to the epigenome. (Art by **Ting Wang**)

CHAPTER 1

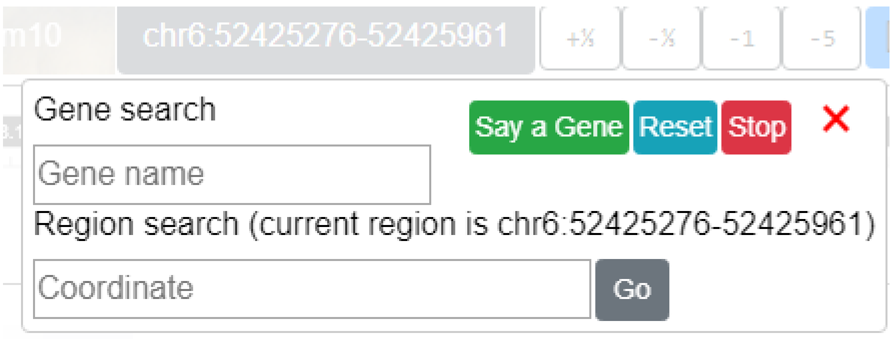
Use the Browser

1.1 The Top Menu



The top navigation menu (above) controls most of the browser functionality. From left to right are the browser logo with version information, species and assembly information, genomic region locator, zoom in/out tools, Tracks menu, Apps menu, Settings menu and Documentaion link.

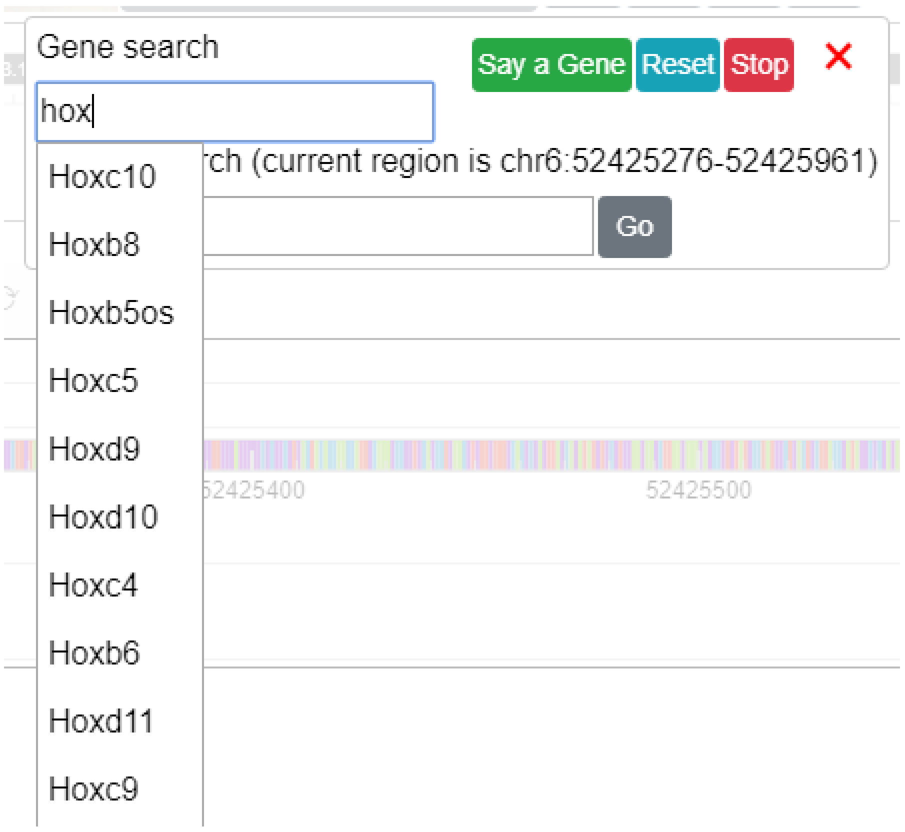
1.2 Genomic Region Locator



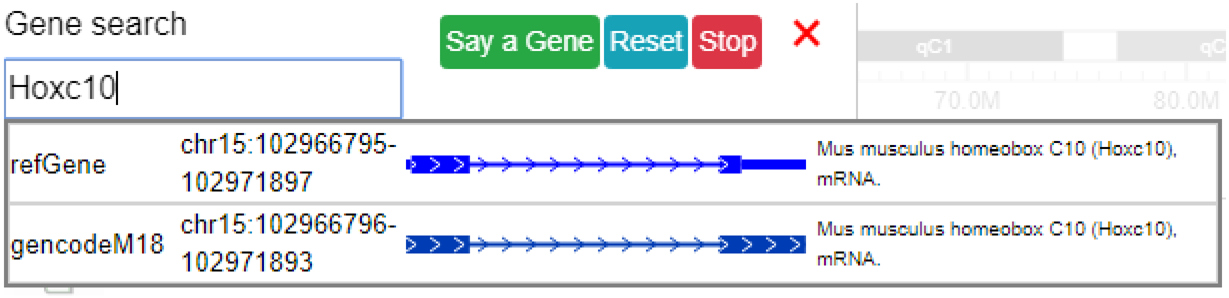
The genomic region locator allows the user to navigate to a region or gene.

1.2.1 Gene Search

You can type a gene name/symbol, like `Hox`, and when the input content reaches 3 characters the browser will try to find gene symbols starting with what you typed:



After a gene is selected a dropdown menu will pop up with isoforms for the gene. After clicking an isoform the browser will navigate you to its genomic region.



refGene	chr15:102966795-102971897	Mus musculus homeobox C10 (Hoxc10), mRNA.
gencodeM18	chr15:102966796-102971893	Mus musculus homeobox C10 (Hoxc10), mRNA.

Voice input gene symbol

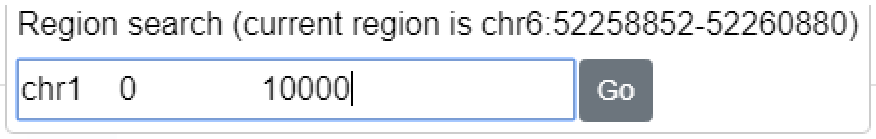
From this set of buttons,

, click the **Say a Gene** button, your web browser will ask you for permission to access your microphone devices, choose *Allow*, and the browser will start to listen to what you are saying. You can start saying letters one by one, like H, O, X, if you click the red **stop** button, what you said "HOX" will populate the gene search box and suggested gene symbols will pop up. As before you can then choose the gene and isoform you want to navigate to.

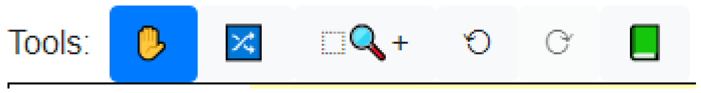
Note: This feature is dependent on web browser support. A web browser without support for speech recognition won't see this UI.

1.2.2 Region Search

Below the gene search box you can use the region search box to navigate to specific genomic coordinates. Formats such as `chr6:52258852-52260880` or `chr6 52258852 52260880` are accepted (the browser is not sensitive to the number of spaces or tabs between the *chr*, *start*, and *stop*.



1.3 Operations in the Tracks View Container

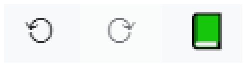


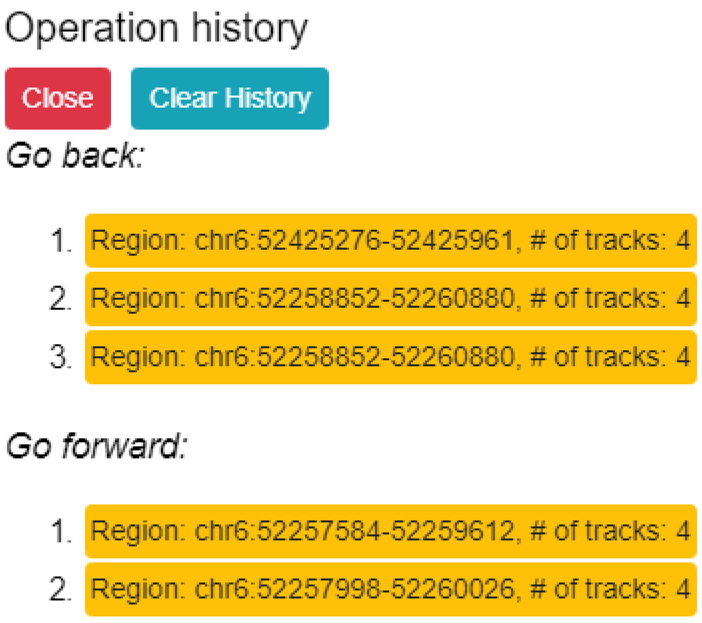
Right above the tracks on the left hand side there are a bunch of tools for operating on the tracks. From left to right these include the move, re-order, zoom, undo, redo, and history tools.

1.3.1 Move/Re-order/Zoom tools

The is the default tool selected by the browser. `Moving` mode allows the user to drag the mouse right or left and the tracks will move along mouse moving to new regions. The allows tracks to be `re-ordered`. The user can drag one or more tracks up or down to change the order of tracks. The allows the user to `zoom in` on a specific region within the current view using the mouse.

1.3.2 Undo/Redo/History tools

 contains the `Undo`, `Redo`, and `History` tools. For instance if you accidently moved to another region and forgot what you were looking at before you can click the `Undo` button to go back. Clicking the `Redo` button allows you to go forward to the step before you clicked `Undo`. The `History` button gives you the 9 most recent operations and allows you to jump to any of these operations or clear the history.


Operation history

Go back:

1. Region: chr6:52425276-52425961, # of tracks: 4
2. Region: chr6:52258852-52260880, # of tracks: 4
3. Region: chr6:52258852-52260880, # of tracks: 4

Go forward:

1. Region: chr6:52257584-52259612, # of tracks: 4
2. Region: chr6:52257998-52260026, # of tracks: 4

1.3.3 Hotkeys

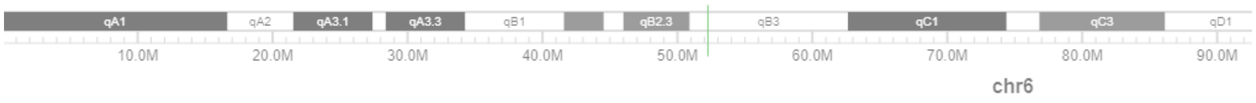
1. `Alt + H` or `Alt + D` for the Drag Tool
2. `Alt + S` or `Alt + R` for the Reorder/Swap Tool
3. `Alt + M` for the Magnify Tool
4. `Alt + Z` and `Alt + X` to pan one full panel left or right.

1.4 Settings

The `Settings` menu controls global settings for the browser.

1.4.1 Toggle display of the Genome Navigator

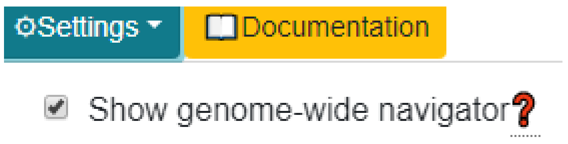
By using the genome navigator (below) users can jump to any genomic region or chromosome(s).



The operations on the genome navigator are:

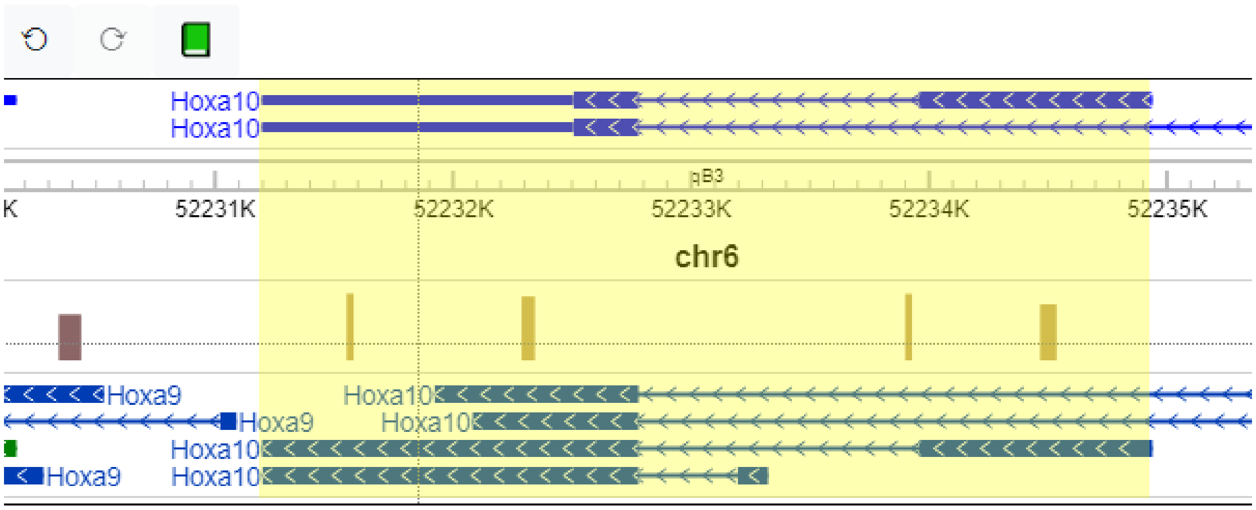
- Left mouse drag: select
- Right mouse drag: pan
- Mousewheel: zoom

The genome navigator can also be hidden to save space when viewing tracks. Click `Settings` on the top menu and uncheck the box to switch off this feature:

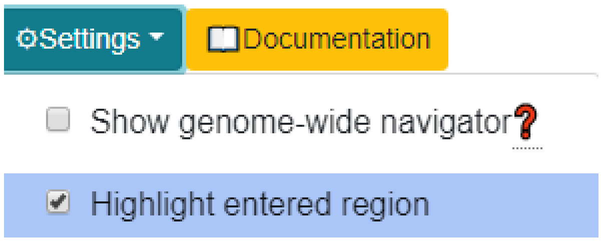


1.4.2 Toggle highlighting of enter region

When a user jumps to a region or gene using the Genomic Region Locator, that region or gene is highlighted with a light yellow box.

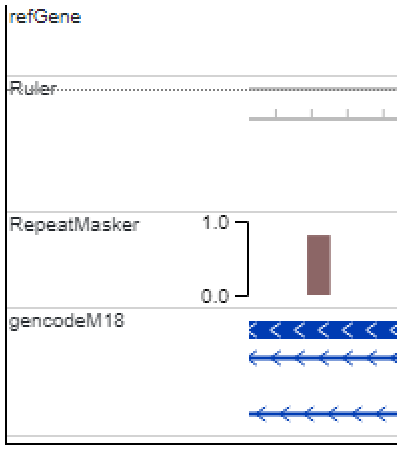


This highlighting can be turned off/on by clicking the botton on the `Settings` menu:

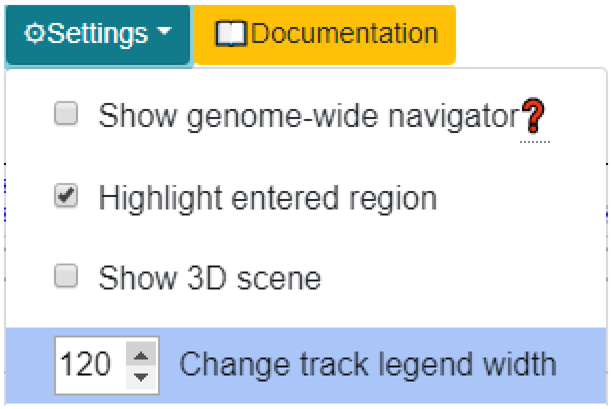


1.4.3 Change track label width

The default width of track labels (below) is 120 pixels.

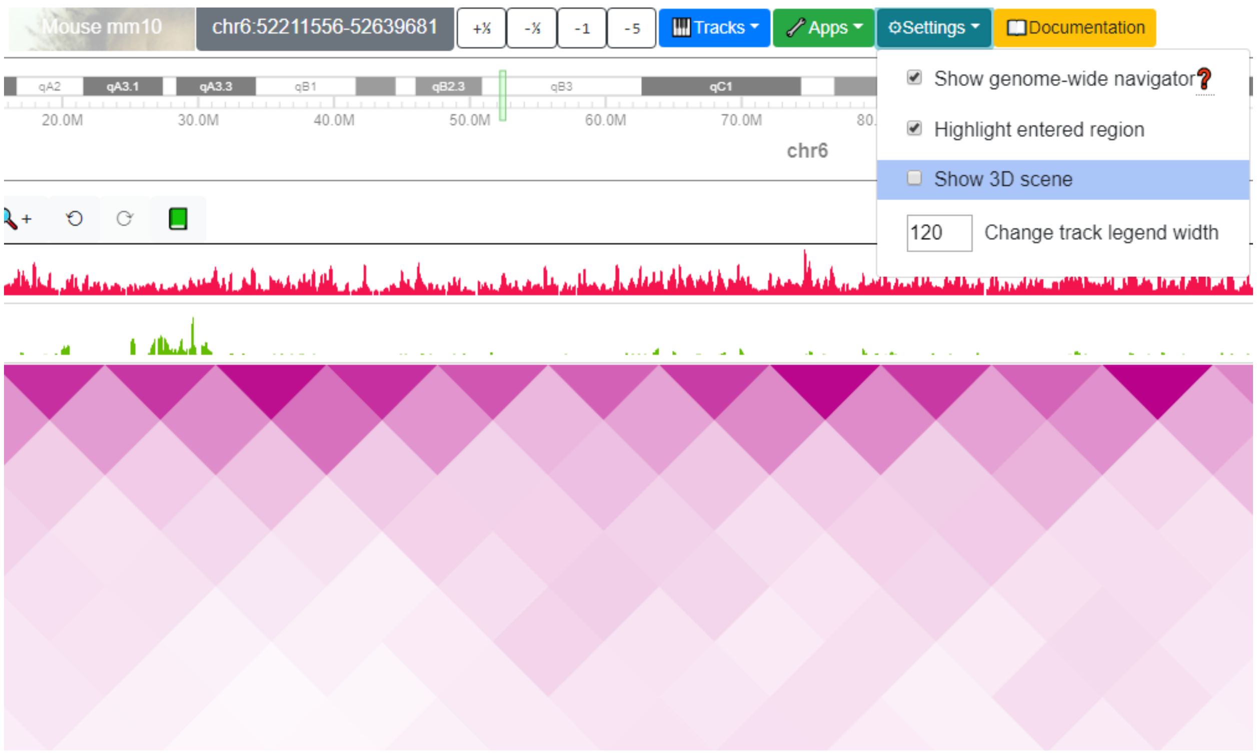


The width of the track label can be configured by the submenu under the `Settings` menu:

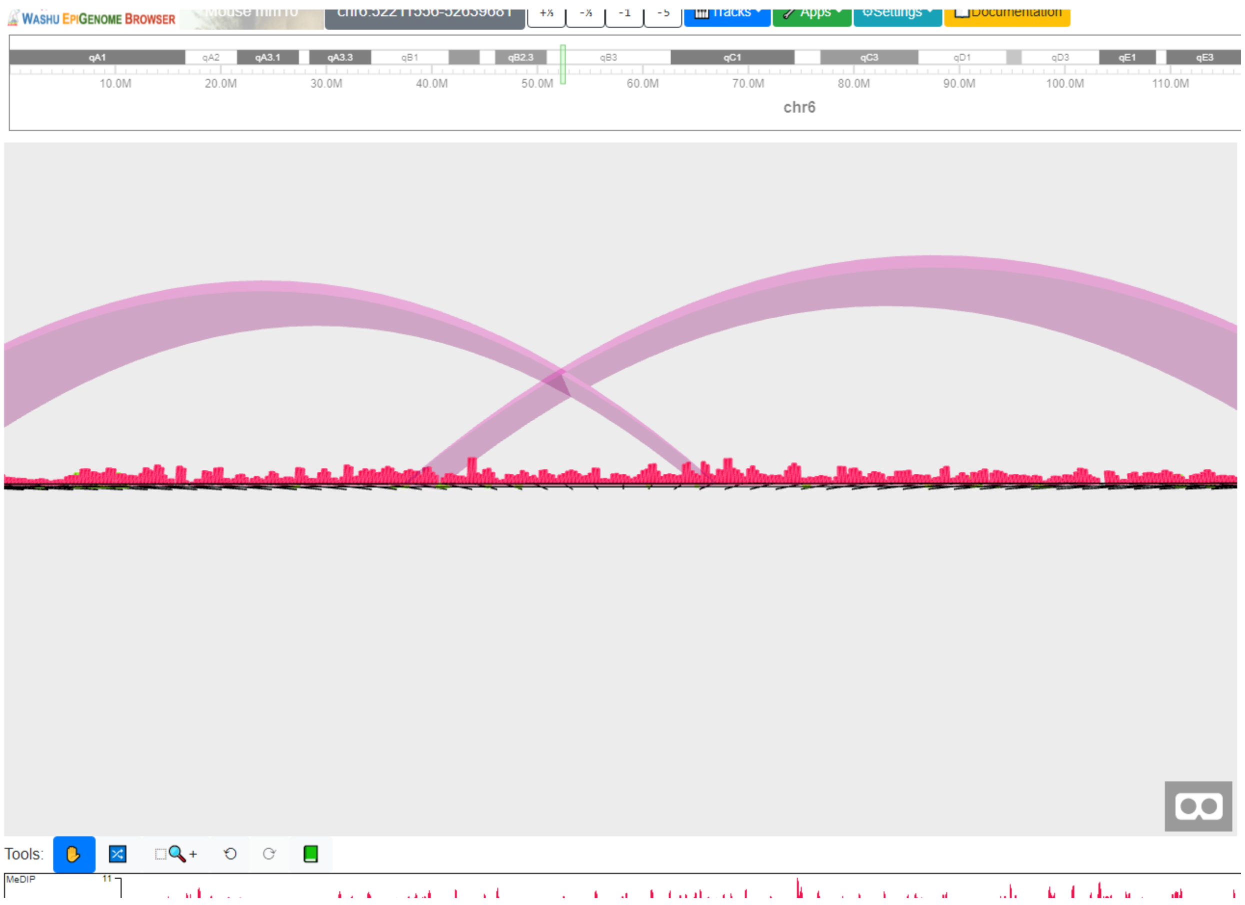


1.4.4 Toggle display of VR mode

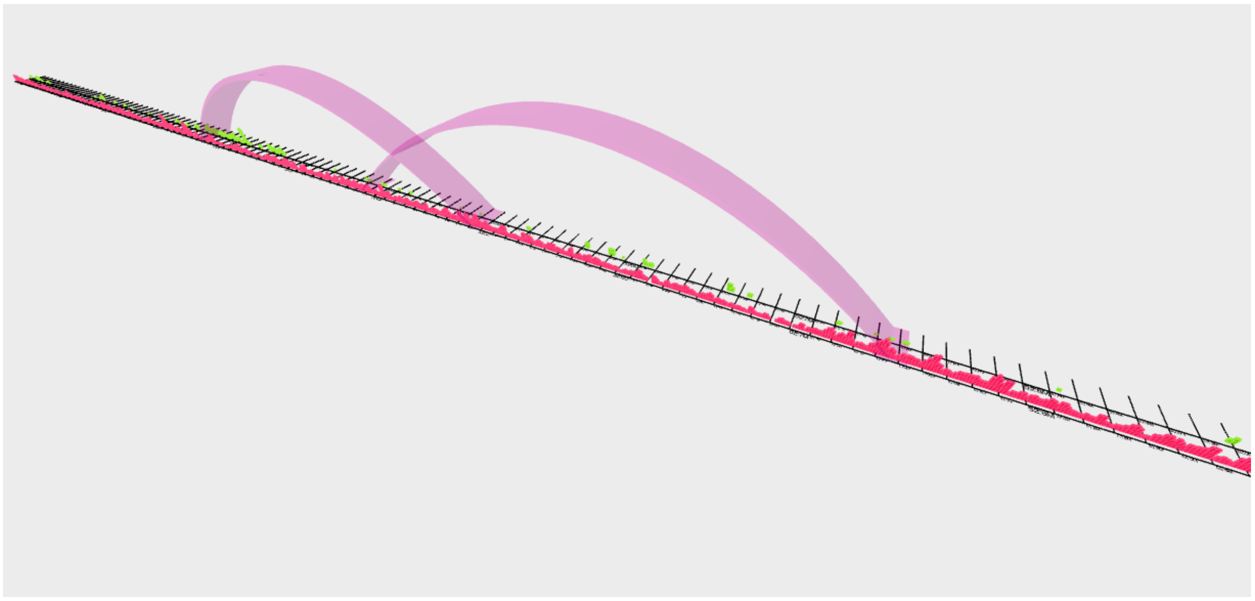
From the `Settings` menu the user can choose to toggle the VR display mode of tracks:



After choose the **Show 3D scene** submenu, a new container with VR view of the tracks will appear:



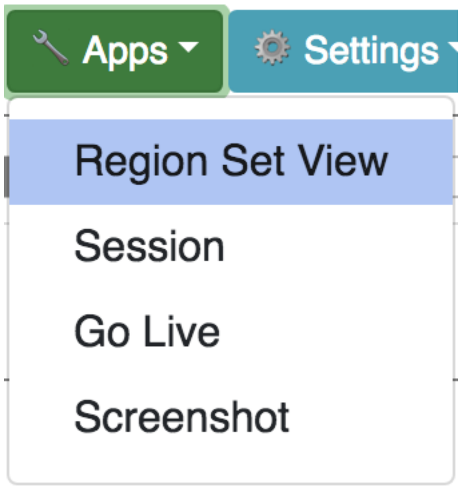
You can click the icon at the bottom right to toggle the full screen display of VR mode, then you can use your mouse and keys `W`, `A`, `S` and `D` to control the view of VR mode, like this view below can show you the interaction between two genomic loci and methylation status along this region in a 3D way.



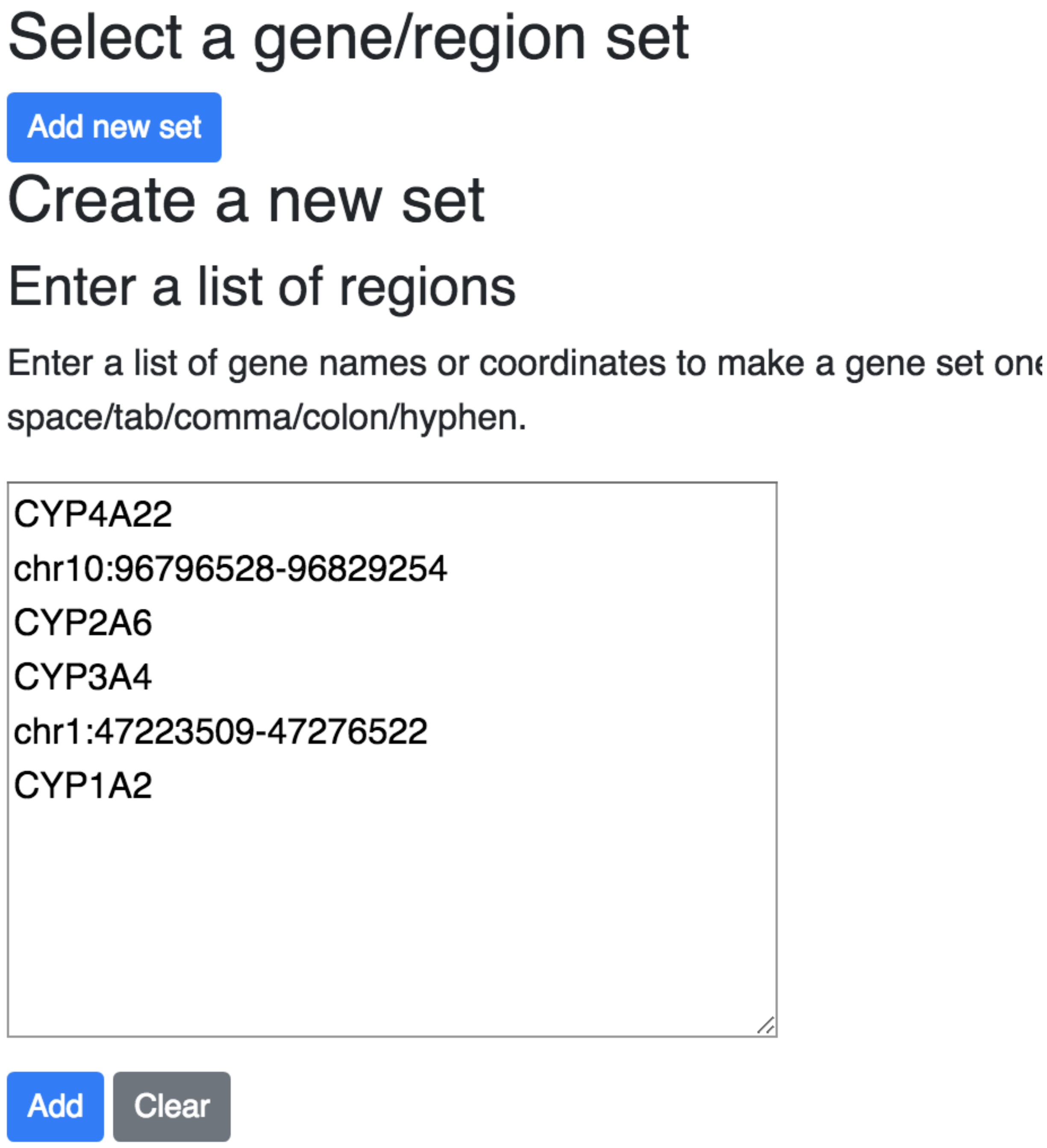
1.5 Apps

1.5.1 Region set view

Users can submit a list of regions or genes to the browser, by choose `Apps` -> `Region set view`:



The brings up the region set user interface, here you can enter a list of gene names or coordinates to make a gene set one item per line. Gene names and coordinates can be mixed for input. Coordinate string must be in the form of "chr1:345-678" fields can be joined by space/tab/comma/colon/hyphen.


Select a gene/region set

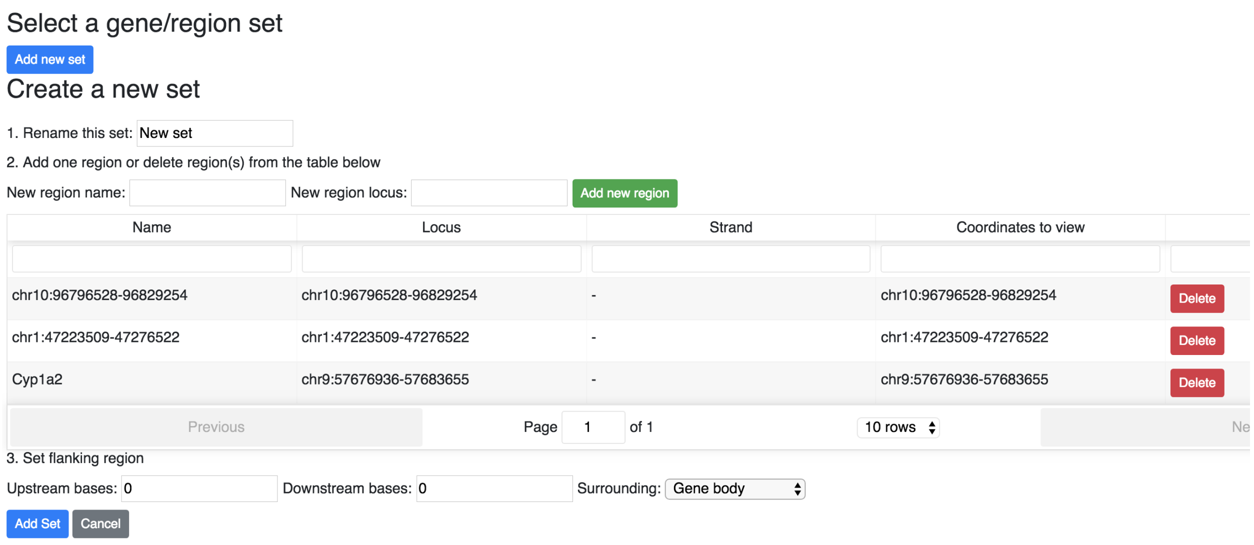
Create a new set

Enter a list of regions

Enter a list of gene names or coordinates to make a gene set one
space/tab/comma/colon/hyphen.

```
CYP4A22
chr10:96796528-96829254
CYP2A6
CYP3A4
chr1:47223509-47276522
CYP1A2
```

After Click the *Add* button, will bring you to the region set editting interface, you can either add region one by one, or delete regions from the table, and set the flanking region strategy:



Select a gene/region set

Add new set

Create a new set

1. Rename this set: New set

2. Add one region or delete region(s) from the table below

New region name: | New region locus: | Add new region

Name	Locus	Strand	Coordinates to view	
chr10:96796528-96829254	chr10:96796528-96829254	-	chr10:96796528-96829254	Delete
chr1:47223509-47276522	chr1:47223509-47276522	-	chr1:47223509-47276522	Delete
Cyp1a2	chr9:57676936-57683655	-	chr9:57676936-57683655	Delete

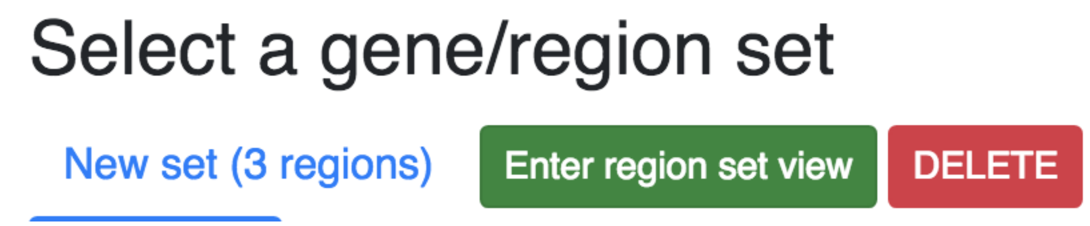
Previous | Page 1 of 1 | 10 rows | Ne

3. Set flanking region

Upstream bases: 0 Downstream bases: 0 Surrounding: Gene body

Add Set | Cancel

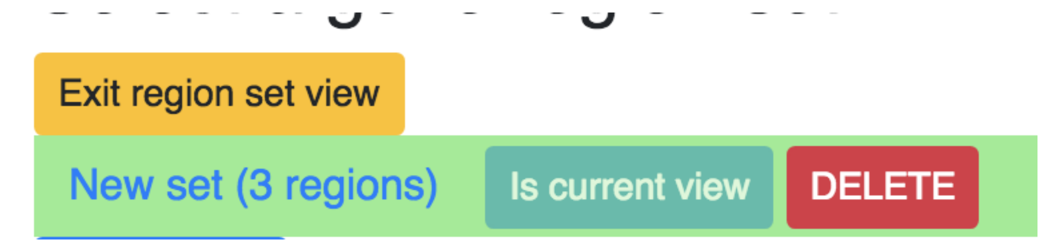
Once you done with edit the set, cick the button *Add set*. Now you have the option to enter regin set view, click the button *Enter region set view*:



Select a gene/region set

New set (3 regions) | Enter region set view | DELETE

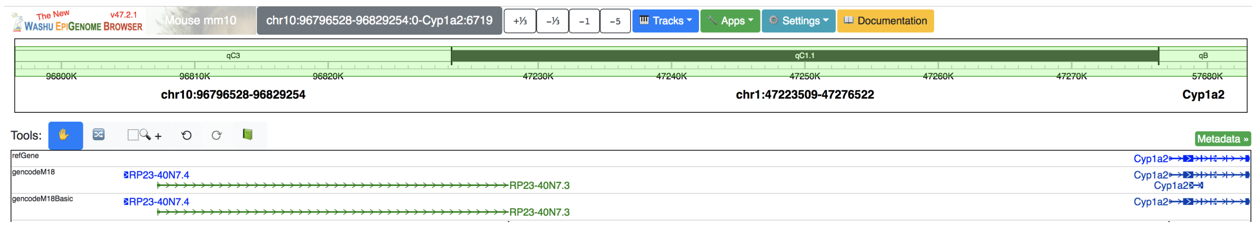
This indicates you are in *region set view* mode and which set you are viewing:



Exit region set view

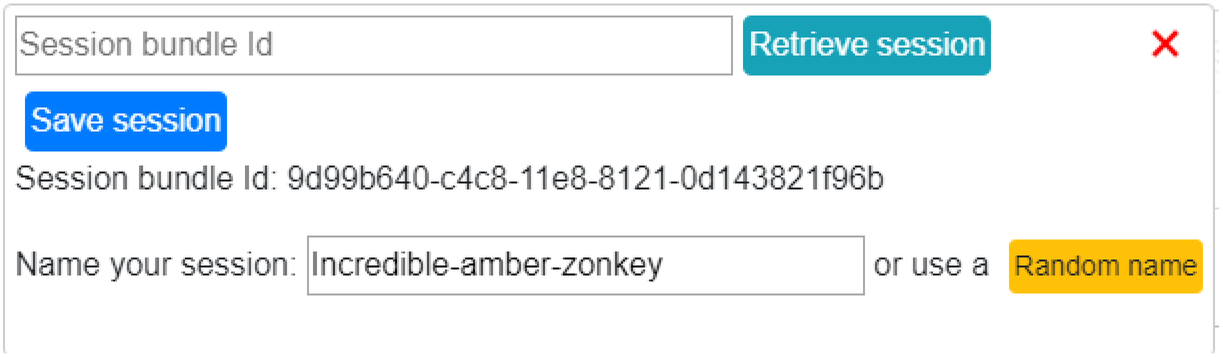
New set (3 regions) | Is current view | DELETE

Go back to the browser, you can your browser view is ordered by your region set:



1.5.2 Session

Choosing `Session` from the `Apps` menu will bring you to the session interface shown below:

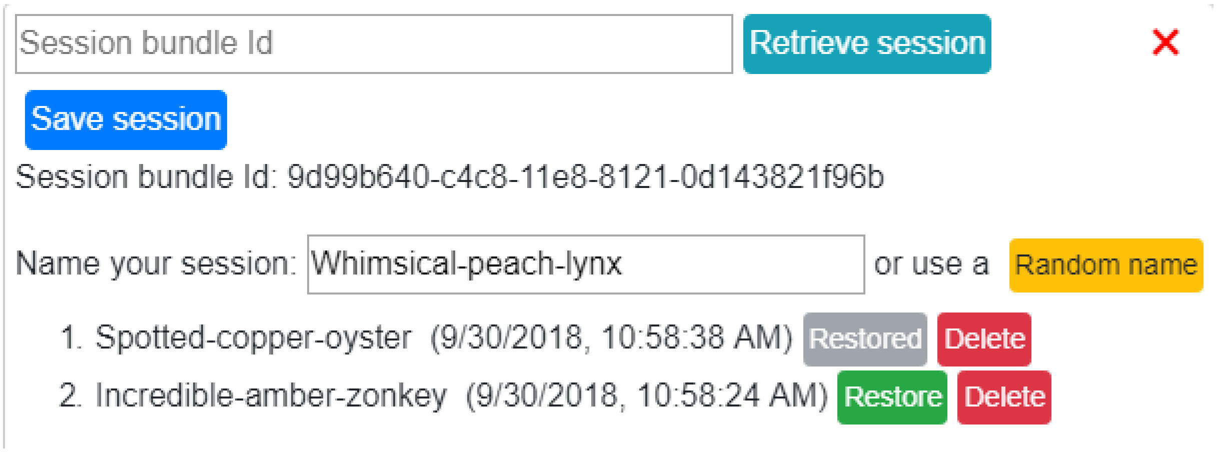


Session bundle Id: 9d99b640-c4c8-11e8-8121-0d143821f96b

Name your session: Incredible-amber-zonkey or use a Random name

Save session

Click the **Save session** button to save a session. A session bundle Id will be created which allows the user to retrieve their session at a later date.



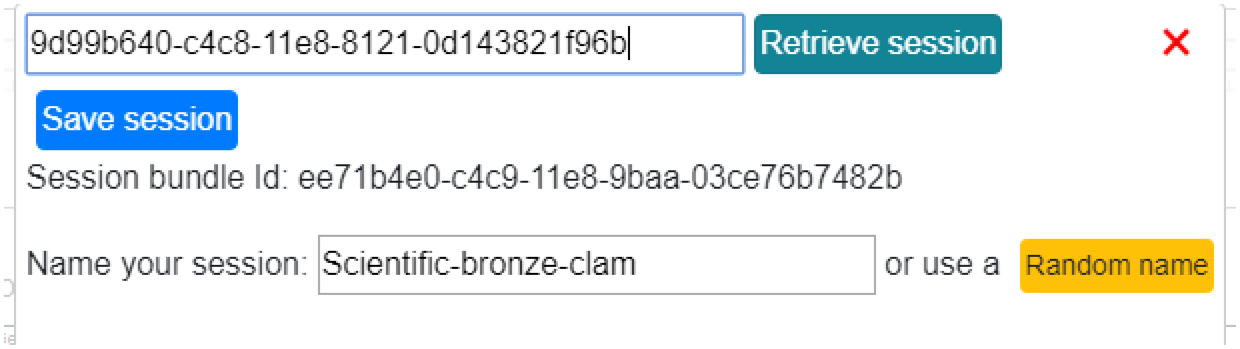
Session bundle Id: 9d99b640-c4c8-11e8-8121-0d143821f96b

Name your session: Whimsical-peach-lynx or use a Random name

1. Spotted-copper-oyster (9/30/2018, 10:58:38 AM) Restored Delete
2. Incredible-amber-zonkey (9/30/2018, 10:58:24 AM) Restore Delete

Retrieve session

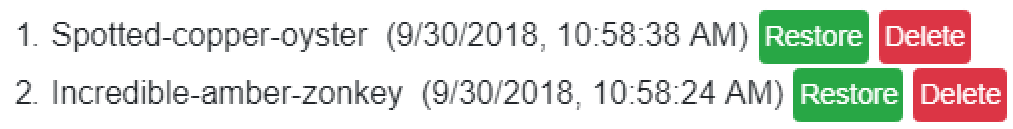
The **session bundle Id** can be used later to retrieve a session by pasting the session bundle id in the session interface and clicking the `Retrieve session` button.


9d99b640-c4c8-11e8-8121-0d143821f96b

Session bundle Id: ee71b4e0-c4c9-11e8-9baa-03ce76b7482b

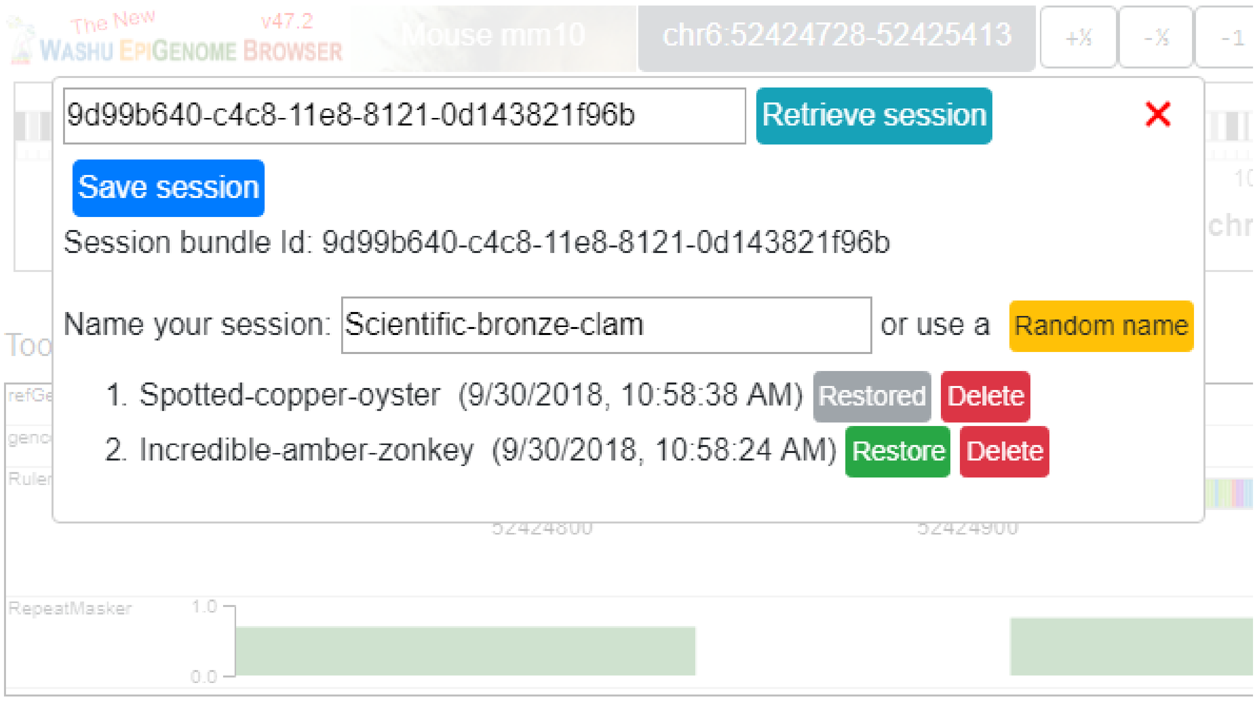
Name your session: Scientific-bronze-clam or use a Random name

Choose which session status you want to restore:



1. Spotted-copper-oyster (9/30/2018, 10:58:38 AM)
2. Incredible-amber-zonkey (9/30/2018, 10:58:24 AM)

Click the green *Restore* button and your session will be restored:



9d99b640-c4c8-11e8-8121-0d143821f96b

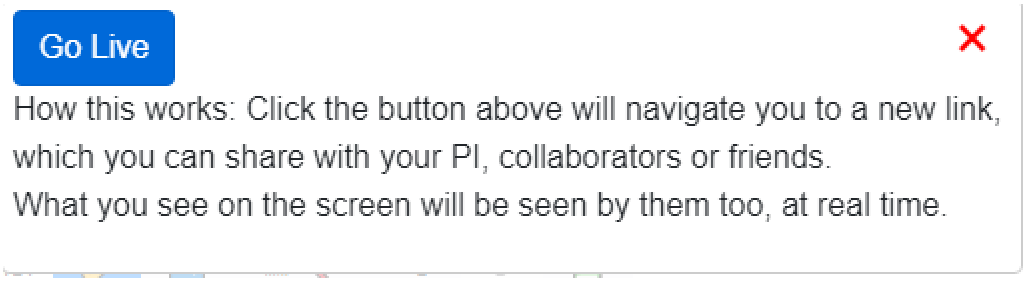
Session bundle Id: 9d99b640-c4c8-11e8-8121-0d143821f96b

Name your session: Scientific-bronze-clam or use a Random name

1. Spotted-copper-oyster (9/30/2018, 10:58:38 AM)
2. Incredible-amber-zonkey (9/30/2018, 10:58:24 AM)

1.5.3 Live browsing

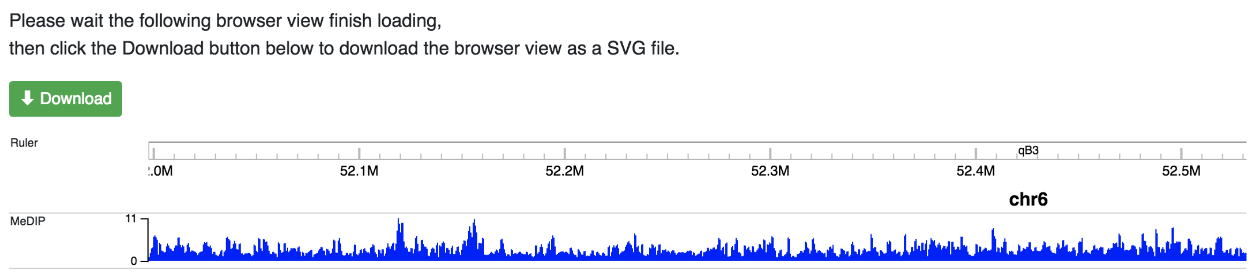
From the Apps menu choose **Go Live**, the browser will navigate you to a new link which you can share with someone else, like your collaborator, your PI, or your friends. Whatever operations are done by you are mirrored on the displays of the people who opened the same link.



How this works: Click the button above will navigate you to a new link, which you can share with your PI, collaborators or friends.
What you see on the screen will be seen by them too, at real time.

1.5.4 Screenshot

Users can create publication quality images using the *Screenshot* tool from the Apps menu. Click the *Screenshot* button and a new window will po pup that re-renders all your tracks as a new SVG file. Once rendered you can click the green download button to save the current browser view as a SVG image file.

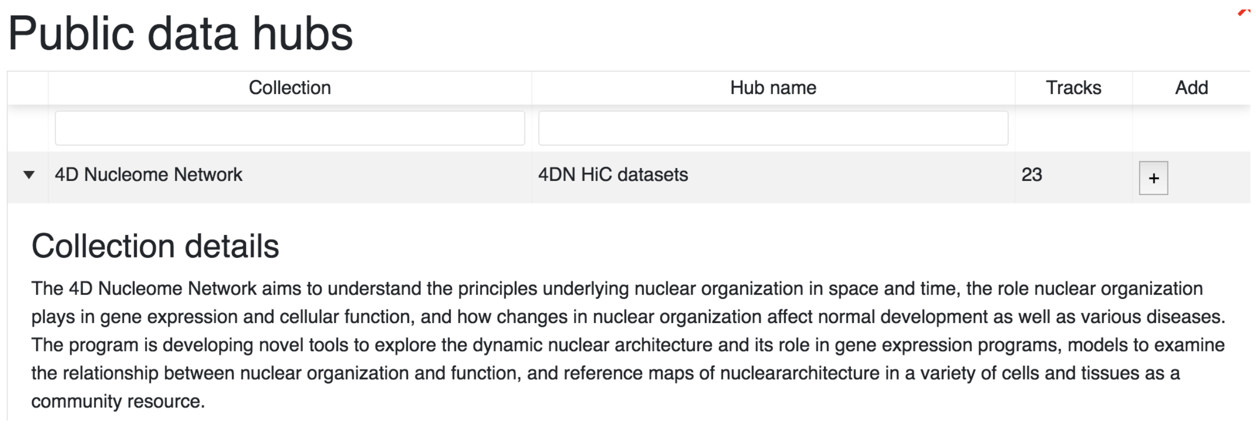

Please wait the following browser view finish loading,
then click the Download button below to download the browser view as a SVG file.

1.6 Track management

The browser collects data from large corsortia like Roadmap Epigenomics, ENCODE, 4DN, TaRGET, etc. The data are called public data/tracks and are organized into different collections called hubs. Along with these public hubs and tracks users can submit their own custom tracks and data hubs to allow for easy comparison.

1.6.1 Add tracks from public hubs

From the `Tracks` menu choose **Public Data Hubs**. This will display all of the public data hubbs available for the species and build you are currently working in. For example, using mouse mm10 annotation the *4D Nucleome Network* hub is available. Click the *Add* button to load this hub:

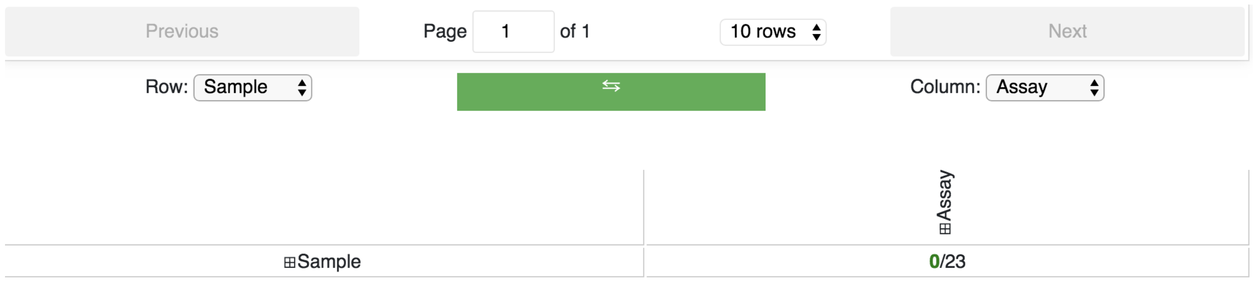

Public data hubs

Collection	Hub name	Tracks	Add
4D Nucleome Network	4DN HiC datasets	23	+

Collection details

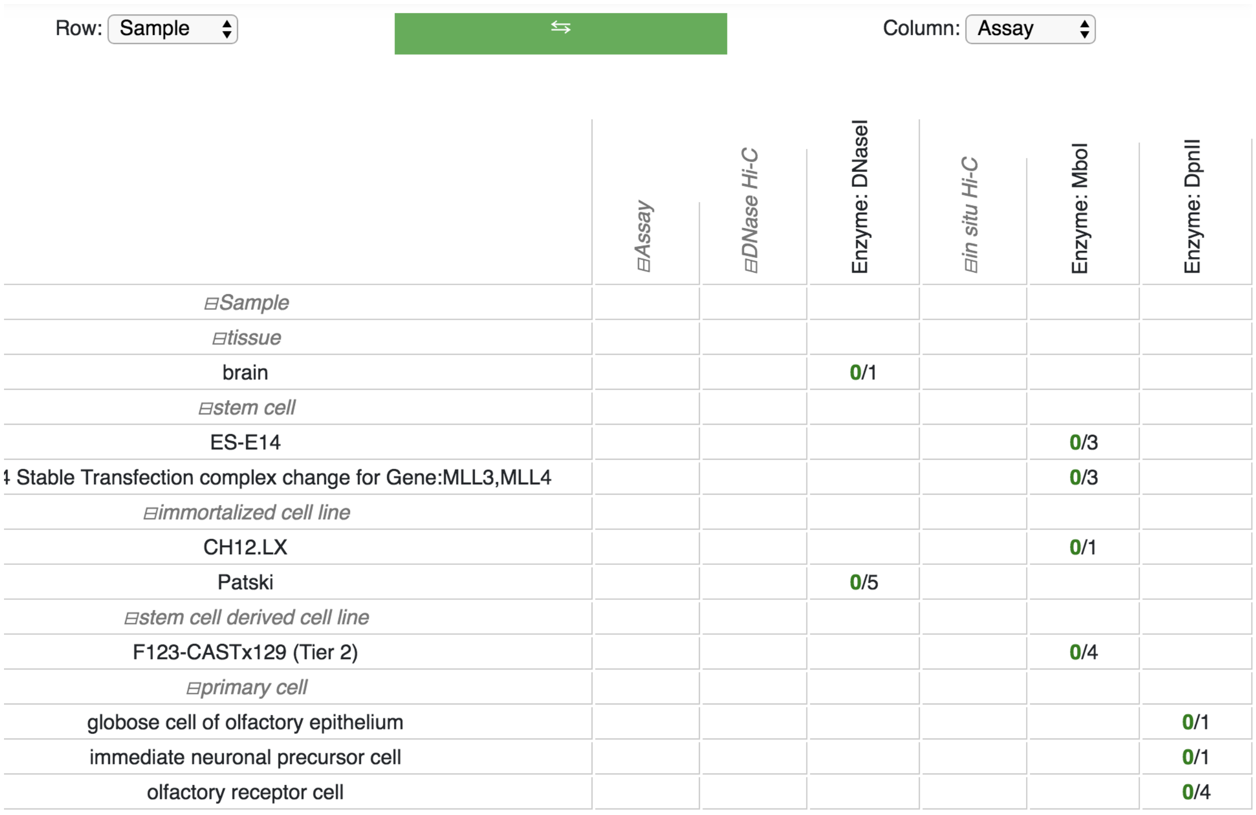
The 4D Nucleome Network aims to understand the principles underlying nuclear organization in space and time, the role nuclear organization plays in gene expression and cellular function, and how changes in nuclear organization affect normal development as well as various diseases. The program is developing novel tools to explore the dynamic nuclear architecture and its role in gene expression programs, models to examine the relationship between nuclear organization and function, and reference maps of nucleararchitecture in a variety of cells and tissues as a community resource.

After a hub is added, a `facet table` containing all tracks will pop up. This allows you to choose any tracks you are interested in. The `facet table` can also be revisted through the menu when you choose **Track Facet Table**:



	⊞Assay
⊞Sample	0/23

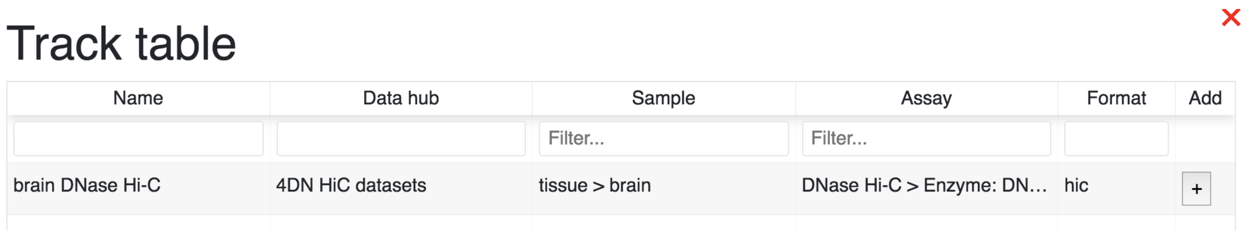
You can expand the row and/or column selection by clicking the + buttons. Row and column displays can also be easily swapped:



Row: Sample ⇆ Column: Assay

	⊟Assay	⊟DNase Hi-C	Enzyme: DNaseI	⊟in situ Hi-C	Enzyme: MboI	Enzyme: DpnII
⊟Sample						
⊟tissue						
brain			0/1			
⊟stem cell						
ES-E14					0/3	
Stable Transfection complex change for Gene:MLL3,MLL4					0/3	
⊟immortalized cell line						
CH12.LX					0/1	
Patski			0/5			
⊟stem cell derived cell line						
F123-CASTx129 (Tier 2)					0/4	
⊟primary cell						
globose cell of olfactory epithelium						0/1
immediate neuronal precursor cell						0/1
olfactory receptor cell						0/4

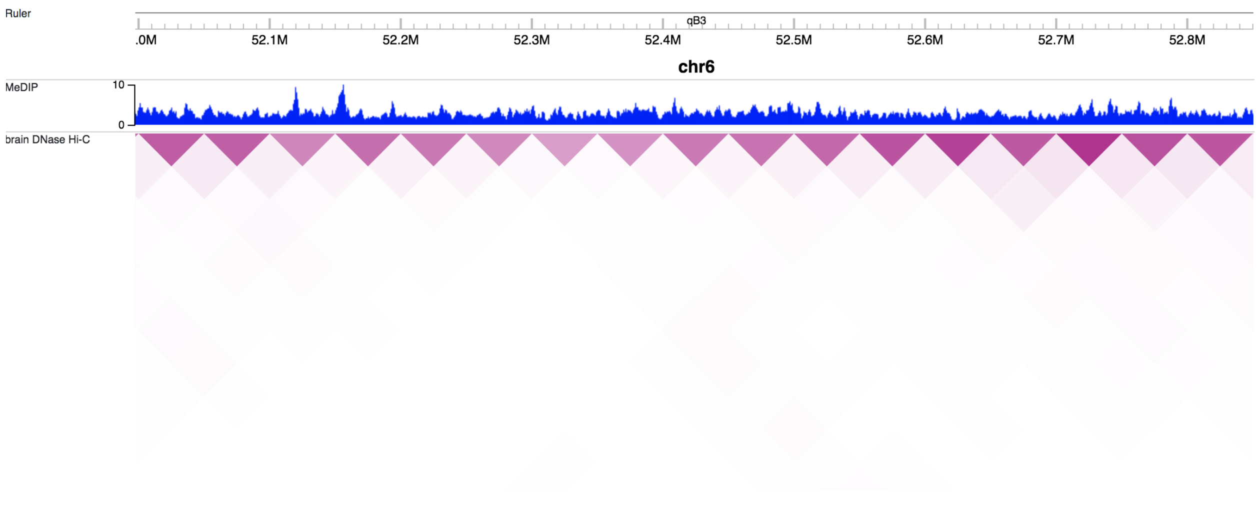
Clicking a cell within the facet table will pop up a new window containing a table with the tacks that match the row and column selections:



Track table

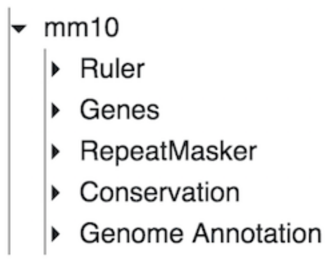
Name	Data hub	Sample	Assay	Format	Add
		Filter...	Filter...		
brain DNase Hi-C	4DN HiC datasets	tissue > brain	DNase Hi-C > Enzyme: DN…	hic	+

Click the *Add* button to add the track(s) you want. You can then view tracks in the browser view window:

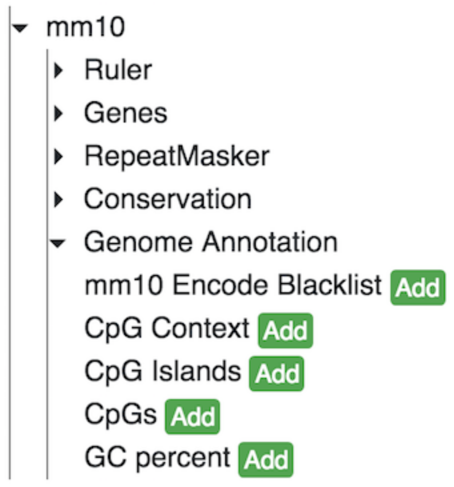


1.6.2 Adding annotation tracks

Users can add numerous annotation tracks from the `Tracks` menu by choosing **Annotation Tracks**.

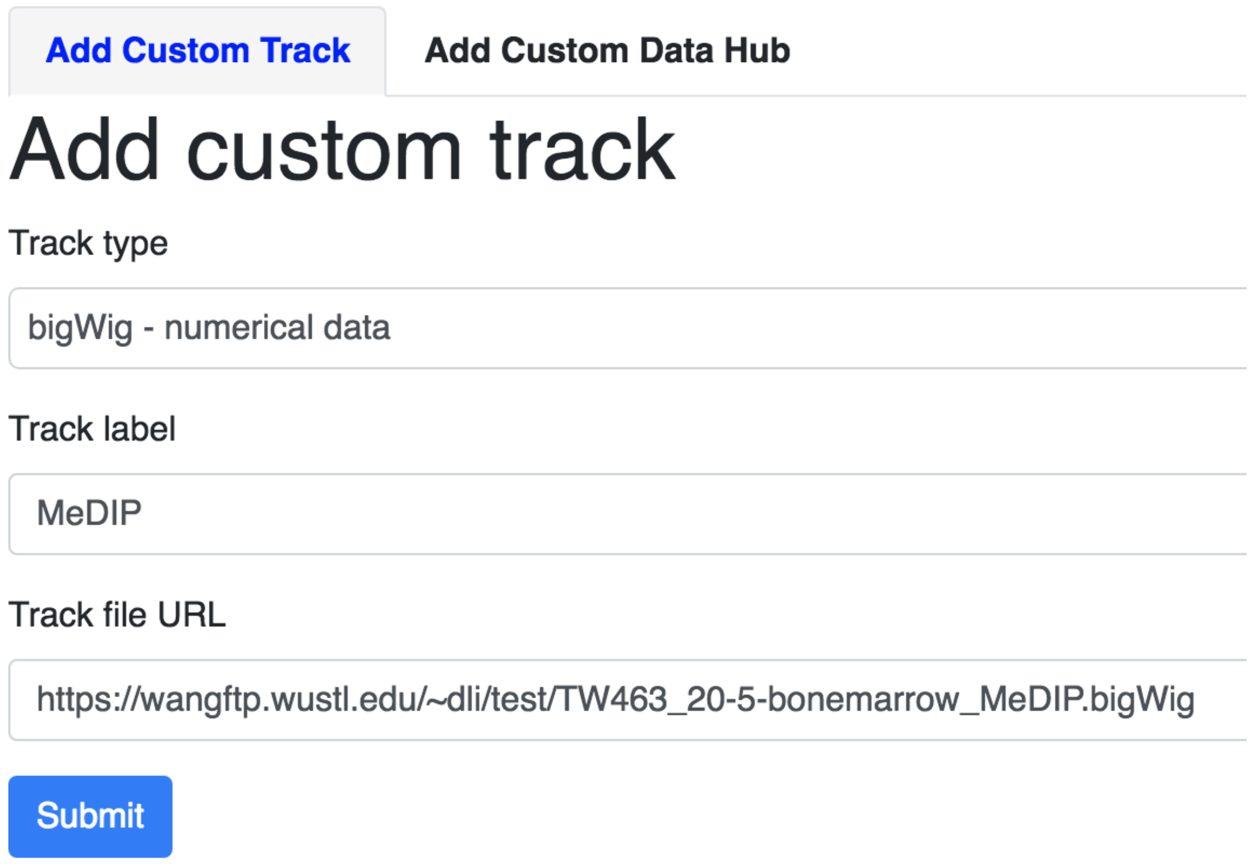


Each header can be expanded to one or more submenus that display tracks that can be added to the browser. The tracks include CpG island information, repeat information, G/C content information, and conservation information to name a few.



1.6.3 Adding a custom track or data hub

Users can also submit their own track as a custom track. For example, say we have a bigWig track located at https://vizhub.wustl.edu/public/tmp/TW463_20-5-bonemarrow_MeDIP.bigWig . From the `Tracks` menu choose **Custom tracks** and a custom track interface will pop up. Fill in the track type, label, and URL before clicking the green *Submit* button:



Add Custom Track | Add Custom Data Hub

Add custom track

Track type

bigWig - numerical data

Track label

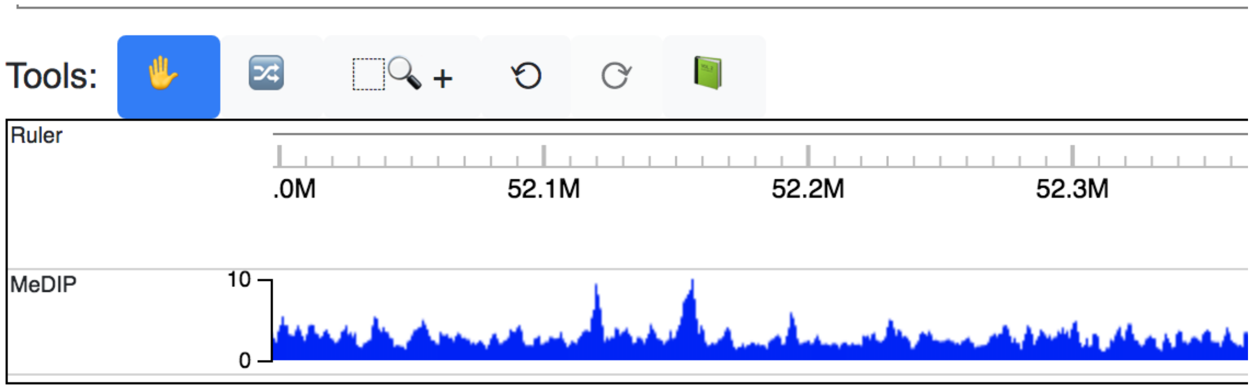
MeDIP

Track file URL

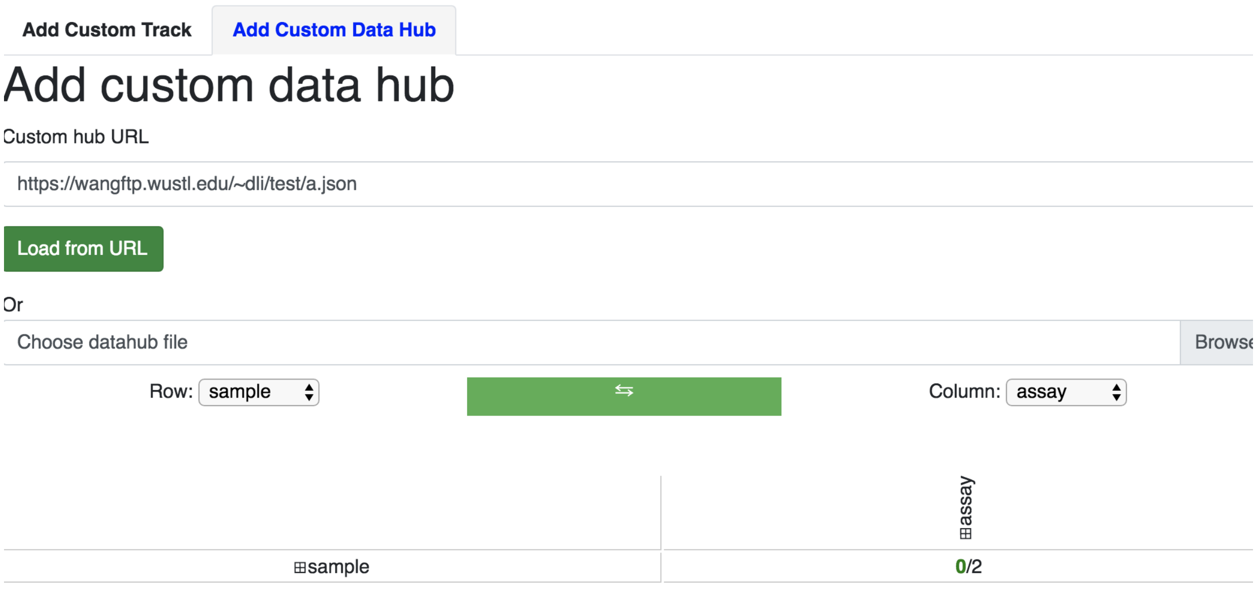
https://wangftp.wustl.edu/~dli/test/TW463_20-5-bonemarrow_MeDIP.bigWig

Submit

You can see the track is added:



Adding a custom data hub is similar to the steps above. For example, say you have a hub located at https://vizhub.wustl.edu/public/tmp/a.json . From the `Tracks` menu choose **Custom tracks**, switch to the *Add custom data hub* tab, paste the URL of your hub, and then click the green *Load From URL* button. from URL.



The tracks within the custom hub can then be added from the generated facet table.

Note: Tracks from custom hubs are hidden by default as users may submit a hub contains hundreds of tracks. Users should add tracks that they want from the facet table.

You can also load a local data hub file in JSON format from your computer using the *file upload* interface, right below the *URL submit* hub interface.

Also see the *Tracks* and *Datahub* sections for how to prepare your tracks and datahub files.

1.7 Track Customization

Tracks can be customized in a multitude of manners.

1.7.1 Selecting Tracks

An indivdual track can be selected by simply right clicking on the tracking on the track. Multiple tracks can be selected by either holding the *shift* button and left clicking on each track or by holding shift and left clicking on a shared metadata term of consecutive tracks. In this manner, multiple tracks can be customized or moved at the same time. To deselect the tracks simply right click and press the button `Deselect # tracks.`

1.7.2 Track Color

Right clicking on `annotation` and `numerical` tracks will display `Primary Color`, `Secondary Color`, and `Background Color` which can all be customized using the color picker. For `methylC` tracks and `categorical` tracks the `Color` and `Background` of each class of elements (e.g. CG, CHG, and CHH) can be personalized. Additionally, for `` methylC``tracks the ``Read depth line color `` can be customized.

1.7.3 Track Height

For each track the height can be customized by right clicking on the track and typing in a number to the panel. At 20 pixels and below for `numerical` tracks the track will display as a heatmap.

1.7.4 Track Display Mode

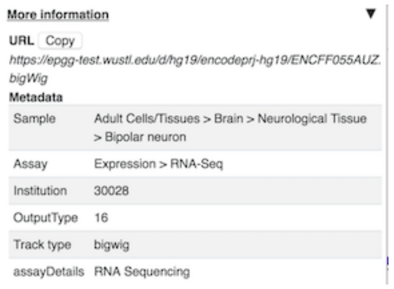
For each `numerical`, `annotation`, or `BAM` track the display can be changed to `DENSITY` or `FULL` mode by right clicking on the track.

1.7.5 Track Y-axis Scale

For each `numerical` track the y-axis can be displayed in `AUTO` or `FIXED` mode by right clicking on the track. The `AUTO` mode will scale the axis based on numerical values in the immediate area of the view range. The `FIXED` mode allows the user to select `a Y-Axis min` or `Y-axis max`. For values above the set max the `Primary color above max` can be set for easy viewing. For values below the set minimum the `Primary color below min` can bet set.

1.7.6 Track Information

If `details` were specified for a track in the data hub file these can be viewed by right clicking on the sample and clicking on the arrow to the right. An easy access `copy` but is also available to copy the `URL` for the track.



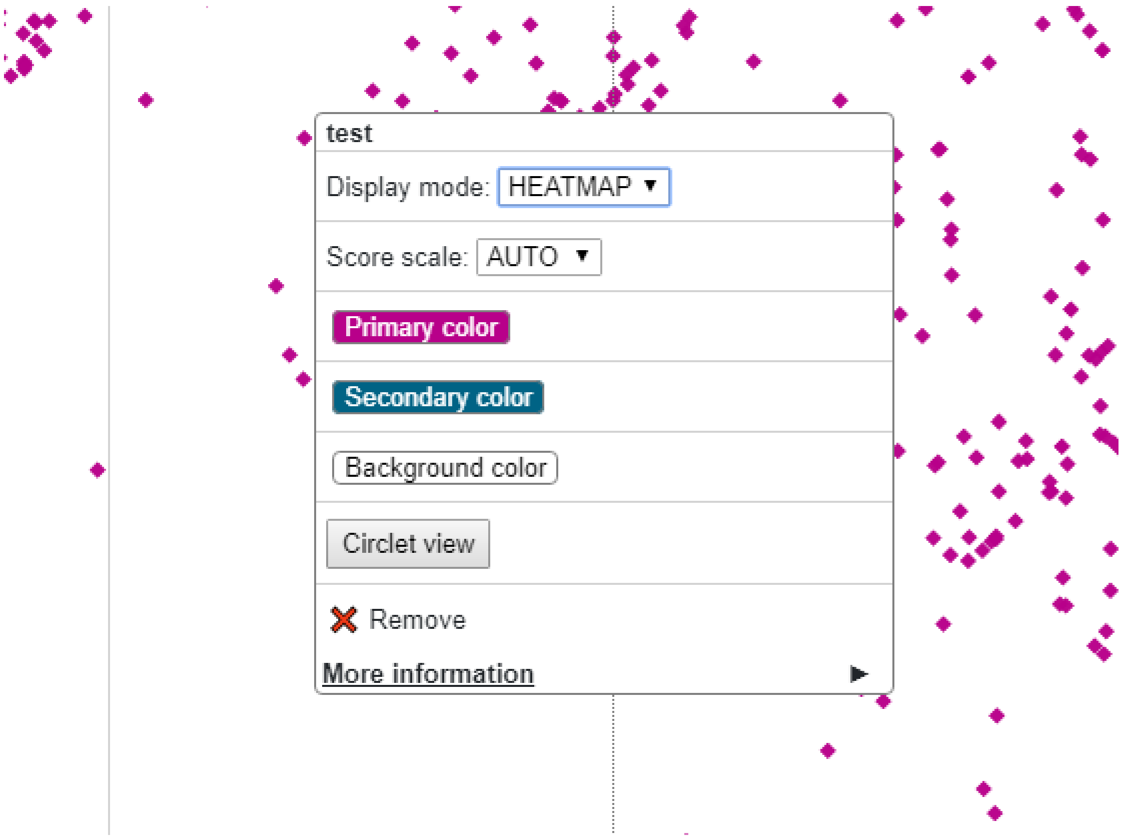
https://epgg-test.wustl.edu/d/hg19/encodeprj-hg19/ENCFF055AUZ.bigWig

Metadata

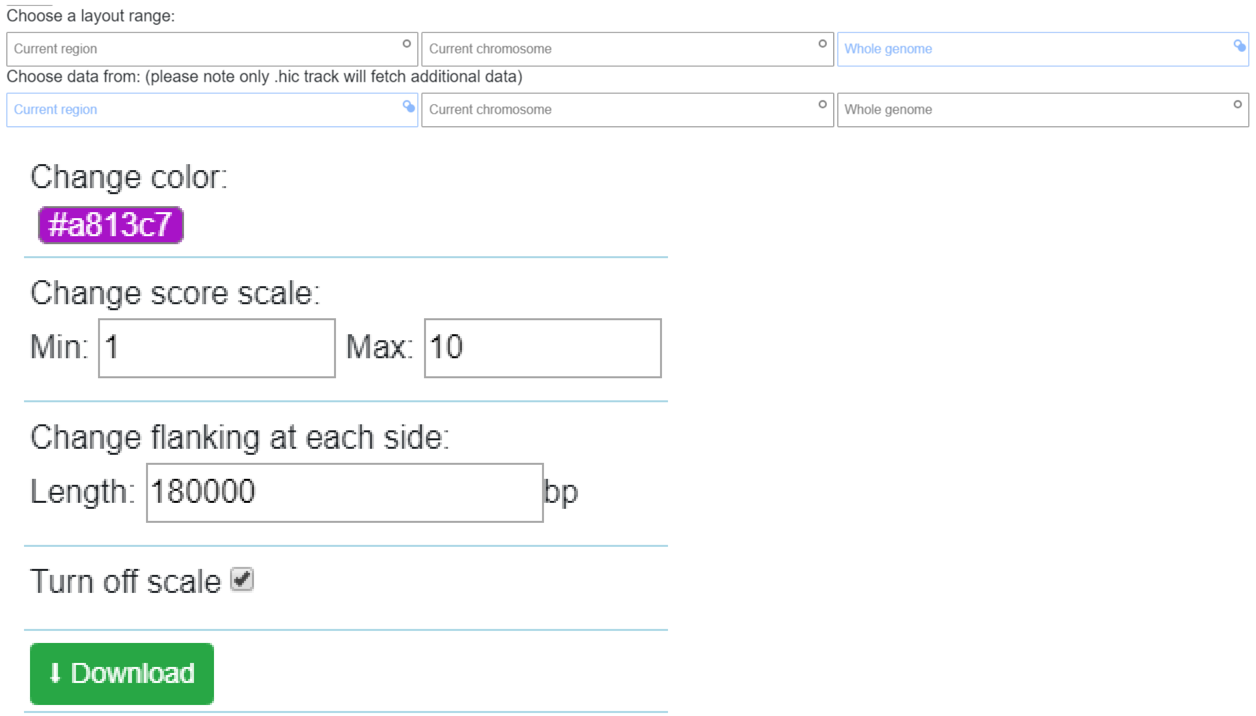
Sample	Adult Cells/Tissues > Brain > Neurological Tissue > Bipolar neuron
Assay	Expression > RNA-Seq
Institution	30028
OutputType	16
Track type	bigwig
assayDetails	RNA Sequencing

1.8 Circlet view for chromatin interaction tracks

For any chromatin interaction track type (*HiC*, *longrange*, *bigInteract*), when you right click the track, you can see the `Circlet view` button:



Click the button will bring you to the **Circlet view** interface. You can config the layout and/or the data source:


Choose a layout range:

Current region ○ Current chromosome ○ Whole genome ●

Choose data from: (please note only .hic track will fetch additional data)

Current region ● Current chromosome ○ Whole genome ○

Change color: #a813c7

Change score scale:

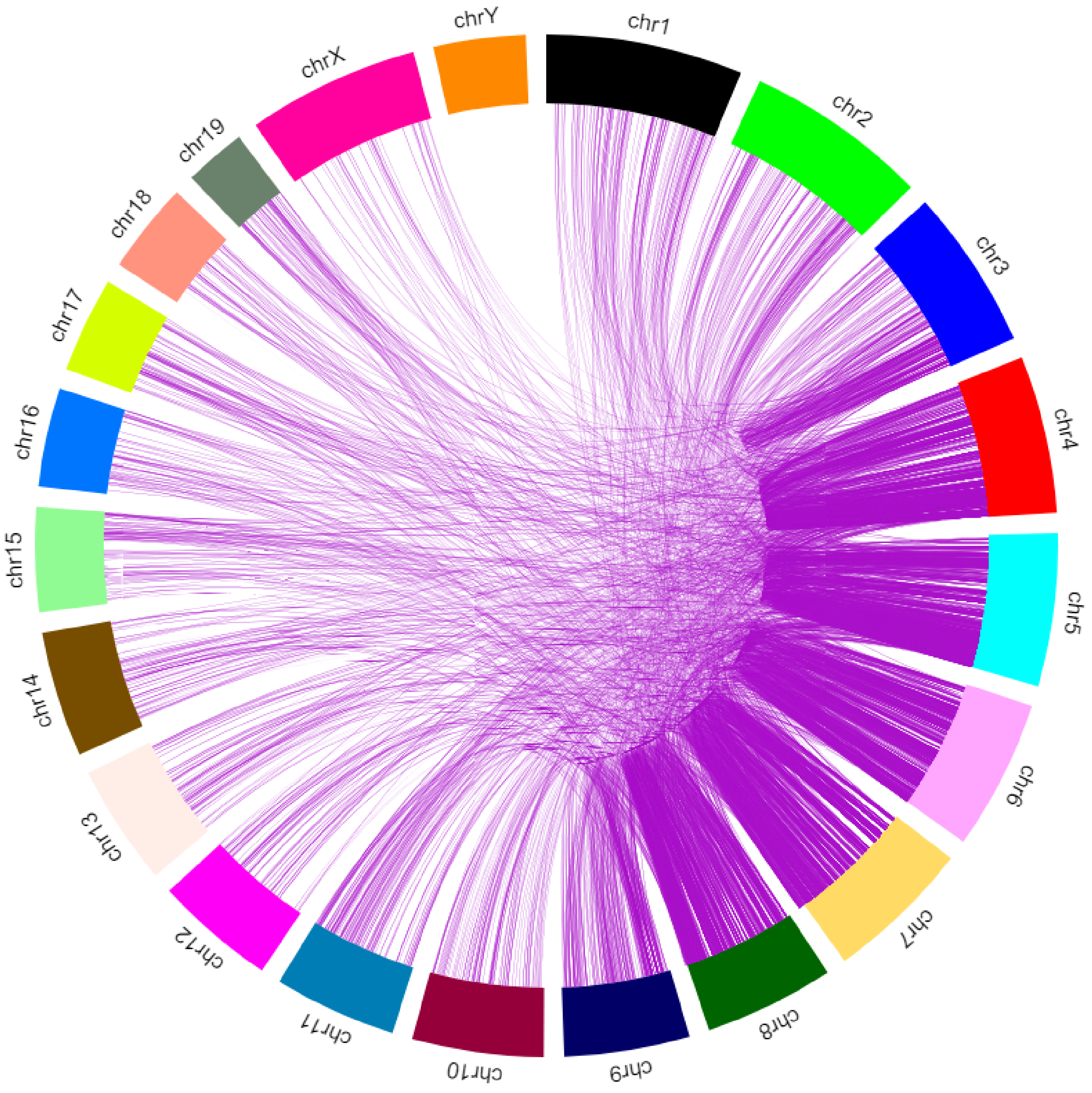
Min: 1 Max: 10

Change flanking at each side:

Length: 180000 bp

Turn off scale ✓

And config the color, scale, flanking region length at each end of one interaction. You also can download the view as a SVG file used for publication.


chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr19
chrX
chrY

CHAPTER 2

Tracks

Important: Since all tracks are hosted on the web with HTTP/HTTPS links provided for submission as tracks, the webservers which are hosting the track files need Cross-Origin Resource Sharing (CORS) enabled.

Quoted from MDN:

```
Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional
HTTP headers to tell a browser to let a web application running at
one origin (domain) have permission to access selected resources
from a server at a different origin. A web application makes a
cross-origin HTTP request when it requests a resource that has
a different origin (domain, protocol, and port) than its own origin.
```

2.1 Configure your webserver to enable CORS

Most likely the browser domain is different from the server the tracks are hosted on. The hosting server needs CORS enabled and for an Apache web server in Ubuntu this setup will work:

```
Header always set Access-Control-Allow-Origin "*"
Header always set Access-Control-Allow-Methods "POST, GET, OPTIONS, DELETE, PUT"
Header always set Access-Control-Max-Age "1000"
Header always set Access-Control-Allow-Headers "x-requested-with, Content-Type,␣
↪origin, authorization, accept, client-security-token"
```

2.2 Prepare track files

The browser accesses track files from their URL. Only a portion of the data, that within the specific view region, are transferred to the browser for visualization. Thus, all the track files need be hosted in a web accssible location using HTTP or HTTPS. The following sections introduce the track types that the browser supports.

Binary track file formats like *bigWig* and *HiC* can be used directly with the browser.

bedGraph, *methylC*, *categorical*, *longrange* and *bed* track files need to be compressed by bgzip and indexed by tabix for use by the browser. The resulting index file with suffix `.tbi` needs to be located at the same URL with the `.gz` file.

Bed like format track files need be sorted before submission. For example, if we have a track file named `track.bedgraph` we can use the generic Linux `sort` command, the `bedSort` tool from UCSC, or the `sort-bed` command from BEDOPS. Here is an example command using each of the three methods:

```
# Using Linux sort
sort -k1,1 -k2,2n track.bedgraph > track.bedgraph.sorted
# Using bedSort
bedSort track.bedgraph track.bedgraph.sorted
# Using sort-bed
sort-bed track.bedgraph > track.bedgraph.sorted
```

Then the file must be compressed using bgzip and indexed using tabix:

```
bgzip track.bedgraph.sorted
tabix -p bed track.bedgraph.sorted.gz
```

Move files “track.bedgraph.sorted.gz” and “track.bedgraph.sorted.gz.tbi” to a web server. The two files must be in the same directory. Obtain the URL to “track.bedgraph.sorted.gz” for submission.

SAM files first need to be compressed to *BAM* files. *BAM* files need to be coordinate sorted and indexed for use by the browser. The resulting index file with suffix `.bai` needs be located at the same URL with the `.bam` file.

Here is an example command:

```
# Using samtools view to convert to bam
samtools view -Sb test.sam > test.bam
# Using samtools sort to coordinate sort the file
samtools sort test.bam > test.sorted.bam
# Using samtools index
samtools index test.sorted.bam
```

2.3 Annotation Tracks

Annotation tracks represent genomic features or intervals across the genome. Popular examples include SNP files, CpG Island files, and blacklisted regions.

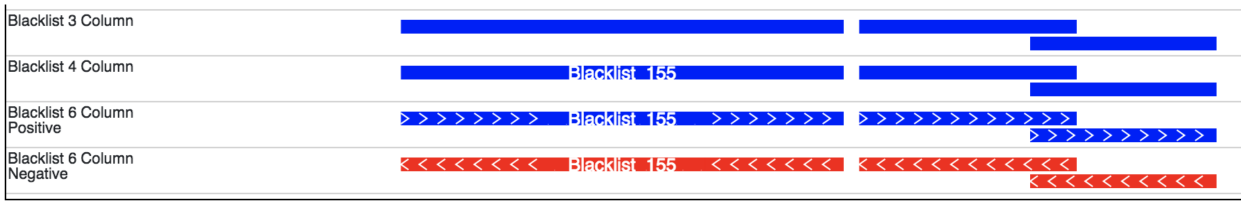
2.3.1 bed

`bed` format files can be used to annotate elements across the genome or to represent reads from a sequencing experiment. For more about the bed format please check the UCSC bed page.

Example lines are below:

```
chr9        3035610 3036180 Blacklist_155   .       +
chr9        3036200 3036480 Blacklist_156   .       +
chr9        3036420 3036660 Blacklist_157   .       +
```

Every line must consist of at least 3 fields separated by the `Tab` delimiter. The required fields from left to right are `chromosome`, `start position` (0-based), and `end position` (not included). A fourth (optional) column is reserved for the name of the interval and the sixth column (optional) is reserved for the strand. All other columns are ignored, but can be present in the file.



Note: The display of a bed file differs by how many columns are provided in the file (see image above). The simplest, 3 column, format just displays blocks for each interval. The four column format displays the name of each element over each interval. If the sixth column is provided in the file then >>> or <<< will be displayed over each interval to represent strand information.

This format needs to be compressed by bgzip and indexed by tabix for submission as a track. See *Prepare track files*.

2.3.2 refbed

The `refbed` format files allows you to upload a custom gene annotation track. It is similar to the refGene bed-like file downloaded from UCSC but with slight modifications. Each file of this format contains (each column is separated by *Tab*):

> chr, transcript_start, transcript_stop, translation_start, translation_stop, strand, gene_name, transcript_id, type, exon(including UTR bases) starts, exon(including UTR bases) stops, and additional gene info (*optional*)

This format needs to be compressed by bgzip and indexed by tabix for submission as a track. See *Prepare track files*.

Hint: The 9th column contains gene type, but is simplified from the Gencode/Ensembl annotations to coding, pseudo, nonCoding, problem, and other. These classes of gene type are colored differently when the track is displayed on the browser.

Hint: The 10th and 11th columns contain exon starts and ends respectively. Each start or end is seperated by a comma.

For example:

```
start1,start2,start3,start4 stop1,stop2,stop3,stop4
100,120,140,160 110,130,150,170
```

Hint: The 12th column contains extra information. This information can be manually annotated or we suggest using Ensembl Biomart to download paired Transcript stable IDs and Gene descriptions. The information in this column must be seperated by *spaces* and not tabs.

All of the below lines will work for additional information in the 12th column:

```
Gene ID:ENSMUSG00000103482.1 Gene Type:TEC Transcript Type:TEC Additional␣
↪Info:predicted gene, 37999 [Source:MGI Symbol;Acc:MGI:5611227]
Gene ID:ENSMUSG00000103482.1 Gene Type:TEC Transcript Type:TEC
ENSMUSG00000103482.1 TEC
Additional Info:predicted gene, 37999 [Source:MGI Symbol;Acc:MGI:5611227]
My Favorite Gene
```

Here are a few example lines in refbed format from gencode.vM17.annotation.gtf (mouse mm10 format):

```
chr1        24910461        24911659        24910461        24911659        -       ␣
↪RP23-109H7.1    ENSMUST00000187022.1    pseudo  24911220,24910461       24911659,
↪24910681       Gene       ID:ENSMUSG00000100808.1 Gene Type:processed_pseudogene␣
↪Transcript Type:processed_pseudogene Additional Info:predicted gene 28594         ␣
↪[Source:MGI Symbol;Acc:MGI:5579300]
chr1        25203443        25205696        25203443        25205696        -       ␣
↪Adgrb3  ENSMUST00000190202.1    coding  25203443        25205696        Gene      ␣
↪                      ID:ENSMUSG00000033569.17 Gene Type:protein_coding Transcript␣
↪Type:retained_intron Additional Info:adhesion G protein-coupled receptor     B3␣
↪[Source:MGI Symbol;Acc:MGI:2441837]
chr1        25276404        25277954        25276404        25277954        -       ␣
↪RP23-21P2.4     ENSMUST00000193138.1    problem 25276404        25277954       ␣
↪Gene                       ID:ENSMUSG00000104257.1 Gene Type:TEC Transcript␣
↪Type:TEC Additional Info:predicted gene, 20172 [Source:MGI Symbol;Acc:MGI:5012357]
chr1        26566833        26566938        26566833        26566938        +       ␣
↪Gm24064 ENSMUST00000157486.1    nonCoding       26566833        26566938       ␣
↪Gene                         ID:ENSMUSG00000088111.1 Gene Type:snoRNA Transcript␣
↪Type:snoRNA Additional Info:predicted gene, 24064 [Source:MGI                    ␣
↪  Symbol;Acc:MGI:5453841]
```

Note: The last optional column is dislayed as a gene description when a gene is clicked on the browser. Our modified format can be easily obtained from available refGene.bed file downloads from UCSC. Gencode GTF and Ensembl GTF files can be manipulated to this format using the Converting_Gencode_or_Ensembl_GTF_to_refBed.bash script in scripts. The script by default puts `Gene ID:`, `Gene Type:`, and `Transcript Type:` in the additional information column. Run with an annotation file, with columns Transcript_ID and Description (seperated by a tab), the script will also add "Additional Info" to the 12th column. The script depends on bedtools, bgzip, and tabix. Lastly, within the script an `awk` array is used to reclassify gene type and can easily be modified for additional gene types.

The script is run as follows:

```
bash Converting_Gencode_or_Ensembl_GTF_to_refBed.bash Ensembl my.gtf my_optional_
↪annotation.txt
bash Converting_Gencode_or_Ensembl_GTF_to_refBed.bash Gencode gencode.vM17.annotation.
↪gtf
bash Converting_Gencode_or_Ensembl_GTF_to_refBed.bash Gencode gencode.vM17.annotation.
↪gtf biomart_2col.txt
```

Warning: Spaces are used as delimiters in the `GTF` files so change gene names with spaces before processing.

For Example:

```
sed -i 's/ (1 of many)/_(1_of_many)/g' Danio_rerio.GRCz10.91.chr.gtf
```

2.4 Numerical Tracks

Currently there are two types of numerical tracks:

- *bigWig*
- *bedGraph*

2.4.1 bigWig

`bigWig` is a popular format to represent numerical values over genomic coordinates. Please check the UCSC bigWig page to learn more about this format.

2.4.2 bedGraph

`bedGraph` format also defines values in diffenent genomic locations. For more about the bedGraph format please check the UCSC bedGraph page.

Example lines are below:

```
chr12   6537598 6537599 28.80914
chr12   6537599 6537600 28.96908
chr12   6537599 6537612 -2
chr12   6537600 6537601 29.30229
```

Every line consists of 4 fields separated by the `Tab` delimiter. The fields from left to right are `chromosome`, `start position` (0-based), `end position` (not included), and `value`.

Note: You can use negative values for reverse strand. Both positive and negative values can exist over the same coordinates (they can overlap). In `bigWig` format negative values can also be specified, but they cannot overlap with positive values.

This format needs to be compressed by bgzip and indexed by tabix for submission as a track. See *Prepare track files*.

2.5 Read Alignment BAM Tracks

2.5.1 BAM

The `BAM` format is a compressed SAM format used to store sequence alignment data. Please check the Samtools Documentation page to learn more about this format and how to manipulate these files.

2.6 Methylation tracks

Methylation experiments like MeDIP-seq or MRE-seq can use *bigWig* or *bedGraph* format for data display. For WGBS if users want to show read depth, methylation context, and methylation level then the data is best suited for the *methylC* format, described below.

2.6.1 methylC

Methylation data are formatted in `methylC` format, which is a 7 column bed format file:

```
chr1    10542   10543   CG      0.923   -       26
chr1    10556   10557   CHH     0.040   -       25
chr1    10562   10563   CG      0.941   +       17
chr1    10563   10564   CG      0.958   -       24
chr1    10564   10565   CHG     0.056   +       18
chr1    10566   10567   CHG     0.045   -       22
chr1    10570   10571   CG      0.870   +       23
chr1    10571   10572   CG      0.913   -       23
```

Each line contains 7 fields separated by Tab. The fields are `chromosome`, `start position` (0-based), `end position` (not included), `methylation context` (CG, CHG, CHG etc.), `methylation value`, `strand`, and `read depth`.

This format needs to be compressed by bgzip and indexed by tabix for submission as a track. See *Prepare track files*.

2.7 Categorical tracks

Categorical tracks represent genomic bins for different categories. The most popular example is the represnetation of chromHMM data which indicates which region is likely an enhancer, likely a promoter, etc. Other uses for the track include the display of different types of methylation (DMRs, DMVs, LMRs, UMRs, etc.) or even peaks colored by tissue type.

2.7.1 categorical

The `categorical` track uses the first three columns of the standard *bed* format (`chromosome`, `start position` (0-based), and `end position` (not included)) with the addition of a 4th column indicating the category type which can be a string or number:

```
chr1    start1  end1    category1
chr2    start2  end2    category2
chr3    start3  end3    category3
chr4    start4  end4    category4
```

Important: when you use numbers like 1, 2 and 3 as category names, in the datahub definition, please use it a string for the `category` attribute in options, see the example below:

```
{
    "type": "categorical",
    "name": "ChromHMM",
    "url": "https://egg.wustl.edu/d/hg19/E017_15_coreMarks_dense.gz",
    "options": {
        "category": {
            "1": {"name": "Active TSS", "color": "#ff0000"},
            "2": {"name": "Flanking Active TSS", "color": "#ff4500"},
            "3": {"name": "Transcr at gene 5' and 3'", "color": "#32cd32"}
        }
    }
}
```

This format needs to be compressed by bgzip and indexed by tabix for submission as a track. See *Prepare track files*.

2.8 Long range chromatin interaction

Long range chromatin interaction data are used to show relationships between genomic regions. *HiC* is used to show the results from a HiC experiment.

2.8.1 HiC

To learn more about the HiC format please check https://github.com/aidenlab/juicer/wiki/Data.

2.8.2 longrange

The `longrange` track is a *bed* format-like file type. Each row contains columns from left to right: `chromosome`, `start position` (0-based), and `end position` (not included), interaction target in this format `chr2:333-444,55`. As an example, interval "chr1:111-222" interacts with interval "chr2:333-444" on a score of 55, we will use following two lines to represent this interaction:

```
chr1    111 222  chr2:333-444,55
chr2    333 444  chr1:111-222,55
```

Important: Be sure to make **TWO** records for a pair of interacting loci, one record for each locus.

This format needs to be compressed by bgzip and indexed by tabix for submission as a track. See *Prepare track files*.

2.8.3 bigInteract

The bigInteract format from UCSC can also be used at the browser, for more details about this format, please check the UCSC bigInteract format page.

2.8.4 cool

Thanks to the higlass team who provides the data API, the browser is able to display cool tracks by using the data uuid from the higlass server, for example, you can use the uuid `Hyc3TZevQVm3FcTAZShLQg` to represent the track for *Aiden et al. (2009) GM06900 HINDIII 1kb*, for a full list of available cool tracks please check http://higlass.io/api/v1/tilesets/?dt=matrix

CHAPTER 3

Datahub

A datahub is a JSON file descibing a set of tracks in the browser. A datahub file is an array of tracks, which are also defined in JSON syntax:

```
[
    {
    "type": "track_type1",
    "name": "track_name1",
    "url": "track_url1",
    "options": {
        "color": "red"
        }
    },
    {
    "type": "track_type2",
    "name": "track_name2",
    "url": "track_url2",
    "options": {
        "color": "blue"
        }
    }
]
```

3.1 Example data hub

Pasted below is an example data hub for mouse mm10:

```
[
    {
    "type": "bigwig",
    "url": "https://vizhub.wustl.edu/public/tmp/TW463_20-5-bonemarrow_MeDIP.bigWig",
    "name": "MeDIP",
    "options": {
```

(continues on next page)

(continued from previous page)

```
        "color": "red",
        "backgroundColor":"#FFE7AB"
        },
    "metadata": {
        "sample": "bone",
        "assay": "MeDIP"
        }
    },
    {
    "type": "bigwig",
    "url": "https://vizhub.wustl.edu/public/tmp/TW551_20-5-bonemarrow_MRE.CpG.bigWig",
    "name": "MRE",
    "options": {
        "color": "blue",
        "backgroundColor":"#C0E3CC"
        },
    "metadata": {
        "sample": "bone",
        "assay": "MRE"
        }
    }
]
```

3.2 Example bigWig track

```
{
    "type": "bigwig",
    "name": "example bigwig",
    "url": "https://vizhub.wustl.edu/hubSample/hg19/GSM429321.bigWig",
    "options": {
        "color": "blue"
    }
}
```

3.3 Example methylC track

```
{
  "type": "methylc",
  "name": "H1",
  "url": "https://vizhub.wustl.edu/public/hg19/methylc2/h1.liftedtohg19.gz",
  "options": {
    "label": "Methylation",
    "colorsForContext": {
      "CG": { "color": "#648bd8", "background": "#d9d9d9" },
      "CHG": { "color": "#ff944d", "background": "#ffe0cc" },
      "CHH": { "color": "#ff00ff", "background": "#ffe5ff" }
    },
    "depthColor": "#01E9FE"
  },
}
```

3.4 Example categorical track

```
{
  "type": "categorical",
  "name": "ChromHMM",
  "url": "https://egg.wustl.edu/d/hg19/E017_15_coreMarks_dense.gz",
  "options": {
      "category": {
          "1": {"name": "Active TSS", "color": "#ff0000"},
          "2": {"name": "Flanking Active TSS", "color": "#ff4500"},
          "3": {"name": "Transcr at gene 5' and 3'", "color": "#32cd32"},
          "4": {"name": "Strong transcription", "color": "#008000"},
          "5": {"name": "Weak transcription", "color": "#006400"},
          "6": {"name": "Genic enhancers", "color": "#c2e105"},
          "7": {"name": "Enhancers", "color": "#ffff00"},
          "8": {"name": "ZNF genes & repeats", "color": "#66cdaa"},
          "9": {"name": "Heterochromatin", "color": "#8    a91d0"},
          "10": {"name": "Bivalent/Poised TSS", "color": "#cd5c5c"},
          "11": {"name": "Flanking Bivalent TSS/Enh", "color": "#e9967a"},
          "12": {"name": "Bivalent Enhancer", "color": "#bdb76b"},
          "13": {"name": "Repressed PolyComb", "color": "#808080"},
          "14": {"name": "Weak Repressed PolyComb", "color": "#c0c0c0"},
          "15": {"name": "Quiescent/Low", "color": "#ffffff"}
      }
  }
}
```

Supported options: *backgroundColor*, *color*, *color2*, *yScale*, *yMax*, and *yMin*.

3.5 Example longrange track

```
{
    "type": "longrange",
    "name": "ES-E14 ChIA-PET",
    "url": "https://egg.wustl.edu/d/mm9/GSE28247_st3c.gz"
}
```

3.6 Example bigInteract track

```
{
    "type": "biginteract",
    "name": "test bigInteract",
    "url": "https://epgg-test.wustl.edu/dli/long-range-test/interactExample3.inter.bb"
}
```

3.7 Example repeatmasker track

```
{
    "type": "repeatmasker",
    "name": "RepeatMasker",
    "url": "https://vizhub.wustl.edu/public/mm10/rmsk16.bb"
}
```

3.8 Example geneAnnotation track

```
{
    "type": "geneAnnotation",
    "name": "refGene",
    "genome": "mm10"
}
```

Note: Please specify the `genome` attibute for gene annotation tracks.

3.9 Example bigbed track

```
{
    "type": "bigbed",
    "name": "test bigbed",
    "url": "https://vizhub.wustl.edu/hubSample/hg19/bigBed1"
}
```

3.10 Example bed track

```
{
    "type": "bed",
    "name": "mm10 bed",
    "url": "https://epgg-test.wustl.edu/d/mm10/mm10_cpgIslands.bed.gz"
}
```

3.11 Example refbed track

```
{
    "type": "refbed",
    "name": "mm10 gencode basic",
    "url": "https://vizhub.wustl.edu/public/tmp/gencodeM18_load_basic_Gene.bed.gz",
    "options": {
            "categoryColors": {
                "coding": "rgb(101,1,168)",
                "nonCoding": "rgb(1,193,75)",
                "pseudo": "rgb(230,0,172)",
                "problem": "rgb(224,2,2)",
```

(continues on next page)

(continued from previous page)

```
                "other":"rgb(128,128,128)"
            }
        }
}
```

Note: `categoryColors` designates colors for the gene type as indicated in the 9th column. The default scheme is shown above for the five classes created by the `Converting_Gencode_or_Ensembl_GTF_to_refBed.bash` script, but any number of categories can be defined.

3.12 Example HiC track

```
{
    "type": "hic",
    "name": "test hic",
    "url": "https://epgg-test.wustl.edu/dli/long-range-test/test.hic",
    "options": {
        "displayMode": "arc"
    }
}
```

3.13 Example cool track

```
{
    "type": "cool",
    "name": "Aiden et al. (2009) GM06900 HINDIII 1kb",
    "url": "Hyc3TZevQVm3FcTAZShLQg",
    "options": {
        "displayMode": "arc"
    }
}
```

Note: please note we are using the uuid `Hyc3TZevQVm3FcTAZShLQg` here from higlass API server instead of a file URL to represent a cool track.

3.14 Example genomealign track

```
{
    "name": "hg19 to mm10 alignment",
    "type": "genomealign",
    "metadata": {
        "genome": "mm10"
    }
}
```

3.15 Example Ruler track

```
{
    "type": "ruler",
    "name": "Ruler"
}
```

3.16 Track properties

3.16.1 type

Requried. `type` specifies the track type, currently supported track types:

- bigWig
- bedGraph
- methylC
- categorical
- hic
- bed
- bigbed
- refbed
- repeatmasker
- geneAnnotation
- genomealign
- longrange
- bigInteract
- ruler

Note: `type` is case insensitive.

3.16.2 name

Requried. `name` specifies the track name used internally by the browser. It is also displayed as the track legend if no *label* speficied. Value can be any string.

3.16.3 label

Optional. `label` specifies the track legend displayed in the browser. It overrides the *name* arrtibute. Value can be any string.

3.16.4 url

Requried. `url` contains the URL to the track file and needs to be HTTP or HTTPS location string.

Important: A `url` is requried for all the tracks in binary format. Gene annotaion tracks, like `refGene`, do not need a `url` as they are stored in the Mongo database. Additional annotation tracks, such as the `ruler` track, also do not need a `url`.

Caution: Each user-provided `url` must link to a publically available website, without password protection, so that the browser can read in the file.

3.16.5 metadata

Optional. An object specifying the metadata of the track.

In this basic example the value of each metadata term is a **string**.

```
"metadata": {
    "sample": "bone",
    "assay": "MRE"
}
```

This example public Roadmap data hub has more complex metadata definitions and makes use of a **list of strings** to build a *hierarchical structure*.

```
{
    "url": "https://egg.wustl.edu/d/hg19/GSM997242_1.bigWig",
    "metadata": {
        "Sample": [
            "Adult Cells/Tissues",
            "Blood",
            "Other blood cells",
            "CD4+_CD25-_Th_Primary_Cells"
        ],
        "Donor": [
            "Donor Identifier",
            "Donor_332"
        ],
        "Assay": [
            "Epigenetic Mark",
            "Histone Mark",
            "H3",
            "H3K9",
            "H3K9me3"
        ],
        "Institution": [
            "Broad Institute"
        ]
    },
    "type": "bigwig",
    "options": {
        "color": "rgb(159,0,72)"
    },
```

(continues on next page)

(continued from previous page)

```
    "name": "H3K9me3 of CD4+_CD25-_Th_Primary_Cells"
}
```

The list of metadata is ordered from more generic to more specific and helps build the facet table hierarchy making the **search** and **filter** functions in track table easier.

3.16.6 details

Optional. If you want to add more information for each track then the `details` attribute is helpful. After right clicking on the track you can click **More Information** and see the `details`, `url`, and `metadata` for each track in the dropdown menu.

```
"details": {
    "data source": "Roadmap Project",
    "date collected": "May 7 2016"
}
```

3.16.7 options

Optional. All track render options are placed in an object called `options`. This object can have the following properties:

color

`color` is used to define the color for each track. A color name, RGB values, or hex color code can be used. For more about color name or RGB please see https://www.w3schools.com/css/css_colors.asp.

color2

`color2` is used to define the color for negative values from the track data. The default is the same as *color*.

backgroundColor

`backgroundColor` defines the background color of the track.

height

`height` controls the height of the track which is specified as a number and displayed in *pixels*.

yScale

`yScale` allows you to configure the track's y-scale. Options include *auto* or *fixed*. *auto* sets the y-scale from 0 to the max value of values in the view region for a given track. *fixed* means you can specify the *minimal* and *maximal* value.

yMax

`yMax` is used to define the *maximum* value of a track's y-axis. Value is number.

yMin

`yMin` is used to define the *minimum* value of a track's y-axis. Value is number.

Important: If you need the track to be in *fixed* scale, you need to specify `yScale` to *fixed* besides of set `yMax` and `yMin`.

colorAboveMax

`colorAboveMax` defines the color displayed when a *fixed* *yScale* is used and a value exceeds the *yMax* defined.

color2BelowMin

`color2BelowMin` defines the color displayed when a *fixed* *yScale* is used and a value is below the *yMin* defined.

displayMode

`displayMode` specifies display mode for each tracks. Different tracks have different display modes as listed below.

type	display mode
bigWig	*auto*, *bar*, *heatmap*
bedGraph	*auto*, *bar*, *heatmap*
geneAnnotation	*full*, *density*
HiC	*arc*, *heatmap*
genomealign	*rough*, *fine*

CHAPTER 4

Installation

4.1 Setup

- Install NodeJS from https://nodejs.org/en/

Note: Feel free to use any package manager tool on your system for installation (`brew`, etc.).

4.2 Start the browser

1. Enter the `frontend` directory
2. **Type `npm install` (just for the first time)** This step will install dependent packages.
3. Type `npm start`

That's it! You are done with your mirror site. The browser is now accessible from http://localhost:3000/browser.

4.3 Setup your own backend API (optional)

By default, your local browser mirror site uses our API service at `https://api.epigenomegateway.org/documentation`, while if you find the species or assembly you are interested is not listed by our API, you can either contact us to add it or build your own API. To build your own API, please follow the steps below:

1. Install MongoDB from https://www.mongodb.com/
2. start mongdb server
3. Enter the `backend` directory
4. Type `npm install`

Then prepare your gene annotation files like the ones for `hg19`, `mm10` etc:

1. Make sure MongoDB is running
2. Enter the `backend` directory
3. **Run `npm run setup`** This step will load the gene annotation data to the MongoDB database
4. Type `npm start`

Now your own backend API is running, change `AWS_API` variable to empty string in `GeneSource.js` file. After this you are using your own API for gene annotation tracks and gene search.

CHAPTER 5

Embedding

To embed the browser in any HTML file, create a HTML page with following contents: (the example shows how to embed a mouse browser with 2 bigWig tracks from ENCODE data portal)

```
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="theme-color" content="#000000">
<title>The New WashU Epigenome Browser</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
↪bootstrap.min.css" integrity="sha384-
↪Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
    crossorigin="anonymous">
<script src="https://igv.org/web/release/2.0.1/dist/igv.min.js"></script>
<script src="https://igv.org/web/jb/release/1.0.0/dist/juicebox.min.js"></script>
<script src="https://aframe.io/releases/0.8.0/aframe.min.js"></script>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
↪KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
    crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js
↪" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q
↪"
    crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"␣
↪integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>
<script src="https://unpkg.com/epgg@latest/umd/epgg.js"></script>
<link rel="stylesheet" href="https://unpkg.com/epgg@latest/umd/epgg.css">
</head>
<body>
<noscript>
    You need to enable JavaScript to run this app.
</noscript>
<h1>Embedding test</h1>
```

(continues on next page)

(continued from previous page)

```
<div id="embed" style="width:1000px"></div>
<h2>some other headings</h2>
<script>
    const container = document.getElementById('embed');
    const contents = {
        "genomeName": "mm10",
        "displayRegion": "chr5:51997494-52853744",
        "trackLegendWidth": 120,
        "isShowingNavigator": true,
        "tracks": [
        {
            "type": "geneannotation",
            "name": "refGene",
            "genome": "mm10"
        },
        {
            "type": "geneannotation",
            "name": "gencodeM19Basic",
            "genome": "mm10"
        },
        {
            "type": "ruler",
            "name": "Ruler"
        },
        {
            "type": "bigWig",
            "name": "ChipSeq of Heart",
            "url": "https://www.encodeproject.org/files/ENCFF641FBI/@@download/
↪ENCFF641FBI.bigWig",
            "options": { "color": "red" },
            "metadata": { "Sample": "Heart" }
        },
        {
            "type": "bigWig",
            "name": "ChipSeq of Liver",
            "url": "https://www.encodeproject.org/files/ENCFF555LBI/@@download/
↪ENCFF555LBI.bigWig",
            "options": { "color": "blue" },
            "metadata": { "Sample": "Liver" }
        }
        ],
        "metadataTerms": ["Sample"],
        "regionSets": [],
        "regionSetViewIndex": -1,
    };
    renderBrowserInElement(contents, container);
</script>
</body>
</html>
```

The key API is the function `renderBrowserInElement`, it accepts the contents array as first argument, and container as second argument which is a DOM element.

CHAPTER 6

Frontend code architeture

Note: This section explains how frontend code is organized, intend to be used for development purpose. Regular browser users don't need to care about this section.

6.1 Quick tour

The client code is in the `frontend` folder. Here is a quick tour of `frontend/src`:

- `components`: All React components.
  - `genomeNavigator`: The navigation bar at the top that allows users to navigate
  - `track`: Track-related components
  - `trackManagers`: UI that manages adding tracks
- `dataSources`: API calls, AJAX calls, database connections, etc. that get data to display.
- `model`: Data models.
- `stories`: Stories for Storybook on which unit tests depend.
- `vendor`: 3rd-party libraries that are not in NPM.

6.2 Suggested order of reading

If you plan to understand the app as a whole here is a suggested order to read the code in:

1. `Feature`: A feature or annotation in the genome.
2. `NavigationContext`: A list of `Features` that represent everywhere a user can navigate. If the `Features` are actually entire chromosomes then the user can effectively navigate the whole genome.

3. `DisplayedRegionModel`: An interval in a `NavigationContext`.
4. `App`: The root component of the app.
5. From `App`, descend into interested components.

6.3 Making a new track type

6.3.1 Make a new TrackConfig

Make a new class that extends `TrackConfig`or one of its subclasses. This class packages many essential track characteristics:

- `getComponent()` - Gets the component that renders the main visualizer and legend of the track.
- `getMenuComponents()` - Specifies context menu items in an array of components. You can choose existing ones in the `contextMenu` directory or make new ones.
- `getOptions()` - The visualizer probably renders with default options like a color. This method returns a plain object containing those options.

You do not have to implement these methods immediately as the base `TrackConfig` class provides minimal defaults. Just work on making the browser render *some* temporary placeholder at first.

6.3.2 Specify when to use the TrackConfig

1. Import your new TrackConfig into `trackConfig/getTrackConfig.js`.
2. Add an appropriate entry to `TYPE_NAME_TO_SUBTYPE`, which maps track type name to track renderer.

6.3.3 Write a new track visualizer component (implement `getComponent()`)

1. Make a new component expecting to receive a bunch of props from `TrackContainer`. `Track.js` documents the props to expect.
2. If you need data assume it will come through the `data` prop. We will add data fetch in the next step.
3. Your new component may `render` anything though it is **highly** recommended you render a `<Track>` component, if not one of the more specialized components like `<AnnotationTrack>` or `<NumericalTrack>`. Pass *all* track container props to these sub-components.
4. In addition to track container props you need to provide certain props to these sub-components, all of which the respective files document.
    - For example, `<Track>` requires a legend and visualizer element. Use the track container props, which includes view region and width, to render a visualizer and pass it to `<Track>`.

6.3.4 Add data fetch

Available data sources are in the `dataSources` folder. If none of them fulfill your needs, write a new class that fulfills the interface of `DataSource.js`. More can be found in that file.

How do we give your visualizer data? Higher-order components! `track/commonComponents` contains track-specific HOCs; their names start with `config-` or `with-`.

`configStaticDataSource` requests a callback that returns a `DataSource` and then returns a *function* that wraps React components. After you use this function, a component will automatically receive three props `data`, `isLoading`, and `error`. These update with the browser's current view region. In particular, the HOC guarantees synchronization of the `data` prop with the current view region if `isLoading` is false.

6.3.5 2. Specify context menu components (implement `getMenuComponents()`)

Specify context menu items with an array of components. You can choose existing ones in the `contextMenu` directory or make new ones.

- Make sure the method returns Component *classes*, not component instances.

6.3.6 3. Specify default options

Default option objects look like the `options` prop of `TrackModel` objects. Context menu items will read these options if the track model does not specify them. Make sure these options are consistent with the way you are rendering your track component! The `configOptionMerging` HOC should help with that.

Once you have a default options object, call `setDefaultOptions()` in the constructor of `TrackConfig` to use them.

6.4 Performance tips

Querying the width or height of any element, for example through `clientWidth` or `getBoundingClientRect()` is slow. Such queries take on the order of 2 to 20 ms. While it is fine to do it once or twice, avoid doing it in a loop. Suppose you aim to plot 500 data points on a SVG and for each point you query the SVG's width. That is already a second or more of computation – very noticable to the user!

6.5 React (and other) gotchas

- On Macs, control + click is the same as a right click which fires a `contextmenu` event. Note that `click` events do not fire on `contextmenu` events. The `mousedown` and `mouseup` events will still fire though.
- When using native DOM events they take priority over React events. This is because React waits for events to bubble to the root component before handling them. This can cause undesirable effects: for example, calling `stopPropagation()` on a React event will not actually stop native events. This StackOverflow post may also help if you have propagation problems: https://stackoverflow.com/questions/24415631/reactjs-syntheticevent-stoppropagation-only-works-with-react-events
- React *always* unmounts components if their parents change type. The `Reparentable` component works around this by using app-unique IDs, but it can cause side effects with React's native events. Use with care.
- Webpack does not support circular dependencies, and while compilation may be successful, an import may resolve as `undefined` at runtime.

6.6 Lessons trying to refactor into WebWorkers

1. Data fetch and track display options are intimately related. For example, what if someone wants HiC data and selects the 5KB resolution option?

2. Thus, for each track type, we have one object that gets the track component, default rendering options, and data fetch/processing.
3. Webpack hangs forever if it encounters a cyclic dependency involving a webworker.
4. The code as in (2) causes a cyclic depdendency. This cycle is [config object] –> [data source] –> [worker] –> [track config deserializer] –> [config object]
5. We cannot have our cake and eat it too.

Unfortunately, this means we cannot pipeline all expensive computation in worker context, while also ensuring track component and data source live in the same place.

CHAPTER 7

Contact us

7.1 Source code and issue tracker

You can find source code at our github repo: https://github.com/lidaof/eg-react

If you find any issue or have any question, please submit an issue here: https://github.com/lidaof/eg-react/issues. Thank you.

7.2 Get help on social media

1. Google plus forum: https://goo.gl/ruXs2Y
2. Facebook page: https://www.facebook.com/WashUEpiGenomeBrowser/
3. Twitter: https://twitter.com/wuepgg

CHAPTER 8

Indices and tables

- genindex
- search