User Manual for TersectBrowser+

David Oluwasusi, Tanya Stead, Gregory Lupton, Gabrielle Baumberg
Supervised by: Dr. Tomasz Kurowski

April 21, 2025

1 Overview

TersectBrowser+ is designed to be useful to a range of Users. Hosting features within the same browser allows the conducting of full analyses within the same page, benefitting both a more experienced researcher as well as a novice to the field.

2 Use Cases - Video transcript

2.1 Example 1:

A tomato genetics researcher wishes to identify differences between a domestic tomato genome and a large number of resequenced genomes of wild relatives.

User profile: Extensive knowledge of introgression genetics and in particular the tomato genome and different species. Lacks knowledge of software development and the different file types involved in setting up the software.

1. The researcher reaches out to administration or informatics support to set up an instance of TersectBrowser+ that contains the large resequenced dataset and relevant GTF files, Reference genomes, and Metadata (e.g. geographic location of species) annotations for the difference accessions.

2. The researcher uses a Local Host Web Address to access TersectBrowser+. The default view is a list of all accessions in the dataset, with one chromosome view selected and a set bin size and zoom level.

   At this stage, the researcher can see a heatmap of SNP density across the different accessions, and gain a brief overview of which accessions differ most from the selected reference sequence. The researcher also can see which locations on the chromosome contain the most variation in SNP density.

3. The researcher is particularly interested in a dark section of the heatmap (I.e. high density of SNPs relative to the selected reference accession), that lies across a chromosomal region containing genes involved in heat stress regulation. The researcher uses the click and select feature along the scale bar to bring up a popup message with further options, and selects the "Set as interval" feature.

   At this stage the researcher understands that the view will change so that the selected chromosomal region (containing high density SNPs in certain accessions) will be zoomed to in some way.

   The researcher sees that TersectBrowser+ view zooms to the selected region. The variation in grey bin highlighting is more obvious in this view. The accession names are also zoomed in, and therefore easier to read.

4. The researcher now sees that a particular set of 5 accessions are the darkest on the heatmap. The researcher would like to know if any key genes cover the dark regions for these 5 accessions. They understand that the gene browser containing gene models and the reference genome is accessible by clicking the binoculars button.

   The researcher sees a new view with the heatmap shifted down so there is a panel of the gene browser at the top of the page. The default view of the gene browser panel is the gene models track, and the first two accession vcf files in the resequenced dataset. The researcher clicks the track selecter button to search for the names of the 5 particular accessions.

   The researcher finds that the zoom on the heatmap has a minimum level past which the scroll zoom no longer works. However, the gene browser panel can be zoomed further than this by selecting a region on the scale bar and clicking "Zoom to region".

   This then zooms to a view where the gene models are visible alongside the variants specific to each accession.

5. The researcher finds a gene model that they know to be important for heat stress. Upon clicking this gene in the browser panel,

2.2 Example 2:

A plant breeder intends to gain intellectual property (IP) recognition for a particular trait of their Soybean plant breed, and wants to find quantitative evidence to support their application using the genome of this breed.

User profile: High familiarity with phenotypic traits of different Soybean plant breeds, as well as history of intended Soybean introgressions by different plant breeders. Lacks knowledge of statistical generation of phenetic trees using pairwise relationships, as well as how the data in the TersectBrowser+ is set up to view the whole dataset.

The breeder opens up the phenetic tree view, in order to view a summary of pairwise differences between the different accessions.

The breeder uses the barcode search function to identify whether the sequence within the identified introgression in their plant breed can be used to uniquely identify that breed relative to the rest of the resequenced dataset.

The breeder exports the results of the barcode search to use in their application for IP recognition of their breed.

2.3 Example 3:

A bioinformatics expert intends to learn more about what introgression means in practice as relating to differences between genomes of similar species.

User profile: Expert knowledge in GUI Web-app design, as well as

The bioinformatician can zoom in the view using the plus/minus buttons on the home bar, or using the mouse zoom scroll (up and down).

The bioinformatician uses the introgression prediction tool to identify which high SNP density regions meet the threshold requirements for high likelihood of introgression from a wild cultivar into the domestic cultivar.

3 Admin setup of browser

The Admin is defined as an individual with bioinformatics and software design knowledge, background comprehension for all sections of this Technical Documentation, and acting to facilitate the use of a deployed TersectBrowser+ by the plant breeder main user of the software. When a new dataset is requested for deployment on TersectBrowser+, the admin will need to follow the below steps.

3.1 Download of required material

1. Collect background context from the user.
    - Size and chromosome number of the reference genome.
    - Size and diversity of the resequenced dataset. Any metadata associated with different accessions in the dataset, such as wild variety vs domesticated variety.
    - Whether there are prior identified introgressions known to be functionally relevant for the species. These can be used to 'check' the ability of TersectBrowser+ to identify introgressed regions for the specified dataset.
2. Clone the GitHub repository of TersectBrowser+ to the local machine. Follow the README from Tersect GitHub for issues with install.
3. Download relevant files to to the specified location within the repository: *Tersect-browser/∼/mongo-data/gp_data_copy/*
    - Download all VCF files in the resequenced dataset. Run tabix on these files to create .tbi index files.
    - Download the reference genome FASTA file. Run SAMtools faidx to create .fai index file.

- Download (or generate) an annotation GFF file for the resequenced dataset using SNPeff. If not using SNPeff, ensure that variant impact levels are categorised into 'High', 'Medium', and 'Low' impact. Index the GFF file. Run tabix on this file to create a .tbi index file.

4. Use JBrowse CLI text-index tool to generate Trix text searching files (.ix, .ixx, and .meta.json) from the GFF annotation file.

5. Run the script `add_config_tracks.py` using these downloaded files. This will use a set of jbrowse arguments to load the datafiles as tracks in the Variant Browser component.

6. Run the script `change_dataset_files.py` using these downloaded files as CLI arguments, in order to update the references to these data files in the project.

7. Run the script `add_example_dataset.sh` to prepare the dataset.

4 Config file generation

There are multiple extensions incorporated within TersectBrowser+, and all config files must be correctly generated for full functionality of the browser. However, each extension has different config requirements and may allow limited functionality before the generation of all config files.

4.1 Variant Browser Pane

A config.json file containing a/the genome assembly is first generated.

Using JBrowse CLI

The fasta file must first be indexed as follows (this step can be omitted if a .fai file already exists):

```
$ cd Tersect-browser/~mongo-data/gp_data_copy/
$ samtools faidx genome.fa
```

The config.json file is created and the genome assembly added to it via the following command:

```
$ jbrowse add-assembly <genome.fa> --load copy --out config.json
```

The variant tracks can then be added to the `config.json` file via the following commands. Jbrowse CLI expects the VCFs to be compressed with bgzip and indexed using tabix.

```
$ bgzip file.vcf
$ tabix file.vcf.gz
$ jbrowse add-track file.vcf.gz --load copy --out config.json
```

The generated config file will contain both the `assembly[]` and the `tracks[]`. These can then be manually copied and pasted into their respective assembly.ts and

tracks.ts files - the reference genome fasta will be added to the *extension/genome-browser/src/app/react-components/assembly.ts* file, while the GFF file and all accession VCF tracks will be added to the *extension/genome-browser/src/app/react-components/tracks.ts* file.

Correct config specifications for the reference genome assembly can be found here.

Manual Generation

If Jbrowse CLI is not compatible with the machine, the assembly config can be generated by adding the following text, and changing the parameters trackID, name, and URI where necessary.

```
$ cd extension/genome-browser/src/app/react-components/
$ nano config.json

"name": "SL2.50",
"sequence": {
  "type": "ReferenceSequenceTrack",
  "trackId": "SL2.50-ReferenceSequenceTrack",
  "adapter": {
    "type": "IndexedFastaAdapter",
    "fastaLocation": {
      "uri": "http://localhost:4200/TersectBrowserGP/tbapi/datafi
      "locationType": "UriLocation"
    },
    "faiLocation": {
      "uri": "http://localhost:4200/TersectBrowserGP/tbapi/datafi
      "locationType": "UriLocation"
    } } }
```

The track config can also be generated by adding the following text, and changing the parameters trackID, name, and uri where necessary.

```
{
"type": "VariantTrack",
"trackId": "S.lyc␣LA2706",
"name": "S.lyc␣LA2706",
"adapter": {
    "type": "VcfTabixAdapter",
    "vcfGzLocation": {
        "uri": "http://localhost:4200/TersectBrowserGP/tbapi/data
        "locationType": "UriLocation"
    },
    "index": {
        "location": {
            "uri": "http://localhost:4200/TersectBrowserGP/tbapi/
            "locationType": "UriLocation"
        },
        "indexType": "TBI"
```

```
    }
},
"assemblyNames": [
    "SL2.50"
]       }
```

A script to automate this process (the tracks will be more numerous than assemblies) can be found on Elvis at *gp_data_copy/scripts/generate_track_config.py*, and a script to generate tbi files for all VCF files, *generate_tbi.sh*, is also present in this folder.

4.2 Gene Model Track

Using JBrowse CLI

Once a `config.json` file for the main reference assembly has been created, run the following commands to sort the GFF3 file, create the .tbi file, and add the config files to the `config.json` using these Jbrowse commands:

```
$ jbrowse sort-gff <yourfile.gff> |
    bgzip > <yourfile.sorted.gff.gz>
$ tabix <yourfile.sorted.gff.gz>
$ jbrowse add-track <yourfile.sorted.gff.gz> --load copy
```

This will create a `FeatureTrack` under the `tracks[]` section in the `config.json`. Then copy and paste the `FeatureTrack` to the *extension/genome-browser/src/app/react-components/tracks.ts* file.

Note: the Jbrowse "sort-gff" command just automates the following shell command

```
$ (grep "^#" in.gff; grep -v "^#" in.gff |
    sort -t"`printf␣'\t'`" -k1,1 -k4,4n)
    > sorted.gff;
```

Manual Generation

If Jbrowse CLI cannot be accessed, the `FeatureTrack` can be generated by adding the following text, replacing trackID, name, and uri with the respective inputs:

```
{
    "type": "FeatureTrack",
    "trackId": "ITAG2.4␣Gene␣Models",
    "name": "ITAG2.4␣Gene␣Models",
    "adapter": {
      "type": "Gff3TabixAdapter",
      "gffGzLocation": {
        "uri": "http://localhost:4200/TersectBrowserGP/tbapi/data
        "locationType": "UriLocation"
      },
      "index": {
```

```
      "location": {
        "uri": "http://localhost:4200/TersectBrowserGP/tbapi/da
        "locationType": "UriLocation"
      },
      "indexType": "TBI"
    }
  },
  "assemblyNames": [
    "SL2.50"
  ] }
```


4.3 Feature Search

4.4 Barcode Generation

5 User access of browser