Welcome to the Genomic Resources Browser Tutorials

This document will help you get familiar with the browser environment available on GitHub (and various other repositories in the same account). We'll start with installation (including finding a place to run this if necessary), then move through practical case studies: viewing RNA-seq data, blasting data, and designing CRISPR guides.

Each case study walks through a real-world scenario while highlighting the relevant tools and demonstrating how different components of the platform connect with each other.

Case Study 0: Installation Walkthrough. Here we will install the ecosystem using docker.

Case Study 1: You have received your RNAseq data back, and you would like to view it against the genome, gene models, and developmental expression patterns. Here we get oriented to JBrowse2 and the unique features added in this platform.

Case Study 2: Upon viewing your data, you have found a target gene in the fruit fly. You would like to search for the sequence in the *Clogmia* genome to find any orthologs. Here we will get oriented to the Blast tools on the platform and how they interconnect with other tools.

Case Study 3: Now that you have identified your loci of interest, you would like to develop an sgRNA primer. You need to consider off target hits, developmental expression patterns, and sequence characteristics (homopolymers, GC content, length, etc). Here we get oriented to the modified version of crisprDesignr and how it interconnects with other tools.

Contents

Case Study 0: Installation Walkthrough

Goal: Get the ecosystem running locally with Docker.
Outline:

1. Why Docker?
2. Clone the repository and build container
3. Build the website (optional)
4. Secure the website
5. Troubleshooting Common Issues
6. Wrapping up

Case Study 1: Viewing RNA-seq Data in JBrowse2

Goal: Orient yourself to JBrowse2 and explore expression data tools.
Outline:

1. Introduction and input data
2. Loading data into the browser
3. Load a BAM file
4. Navigating JBrowse2 basics
5. Unique features of this platform
6. Wrapping up

Case Study 2: BLAST Search Within the Genome (and Beyond)

Goal: Use BLAST to find sequence duplicates and orthologs.
Outline:

1. Introduction and input data
2. Using SequenceServer2.0
3. Visualizing results
4. Cross-tool integration
5. Customization Notes

6. **Wrapping Up**

Case Study 3: Designing an sgRNA with crisprDesignr

Goal: Create an sgRNA primer for CRISPR Cas9 while considering off-targets, expression, and sequence context.

Outline:

1. Introduction and Input Data
2. Using CrisprFinder
3. Using CrispDesignr for unprofiled transcripts
4. Using CrisprViewer
5. Cross-tool integration
6. Wrapping Up

Case Study 0: Installation Walkthrough (Docker Setup)

Before we can view genomes, BLAST sequences, or design CRISPR guides, we need to get the platform running. Often this is a difficult process. I firmly believe software choices should not be dictated by whether or not you can install the thing, I have made this installation as straightforward and easy as possible. Like, distilling everything into four commands and less than an hour.

In this tutorial, we'll walk through:

1. Why we're using Docker and what that means for you.
2. Getting the repository code onto your Jetstream2 VM.
3. Building and running the container.
4. Accessing the browser from your web browser.
5. Troubleshooting common errors (permissions, versions, RAM).

By the end, you'll have a working **browser instance** on your Jetstream2 VM, ready for all the later case studies. The installation takes less than an hour, but I'd **allocate ~2 hours** to this tutorial, given the reading, slow machines, slow internet, etc.

1. Why Docker?

Installations are notoriously fraught for bioinformatics programs. Dependencies are missing, you have the wrong libsomethingorother.so3, or errors are spewing forth page by page with no real context as to what to even google. This is kind of a rite of passage for bioinformatics workers, but that trial is not for today. Instead, we have **containers**.

Containers are a wonderful thing that has been around for many decades, but more recently (late 2010s) became popularlized by Docker, Singularity, Aptainer, and various other platforms. Often these are referred to as Docker Containers or just Containers. They are just that – software in tupperware.

The burp-sealed container has everything you need to run the program – meaning, suddenly, there are no dependency issues! Dependencies, versions, everything is contained within the sealed container. They even have a very small, stripped down version of an operating system.

Containers are portable, making them a great way to ship software. I recommend looking at and using biocontainers, which are also available in a conda version. Both conda and containers are doing similar things – they manage all the messy dependency stuff so you can install stuff with the same command every time. Containers are just more… sealed. Conda depends on the host operating system, like a virus. Containers are more bacterial in that they come with their own operational instructions.

These instructions are usually shipped as a literal file of instructions to build the container, essentially a bare genome called a Dockerfile. You can manually type in all the instructions and build the platform

by hand. You can also easily modify the inputs by simply modifying this file (later case studies cover this).

There are downsides to containers as well, one of which is no different than tupperware. They make a great place to forget about checking on them and suddenly you have super fuzzy software that is 5 years out of date and, while it technically will still run usually, it's risky. You can find some truly crusty software still usable in containers, and they are a favorite way to rescue things from massive platform changes (python2 anyone?). Always check the date of the container, food or software.

Another issue is that the commands are not super beginner friendly for docker, and they are even worse for the large-cluster equivalent called Singularity/Aptainer. Even here, I wrapped the docker commands in shell scripts (setup.sh). If you have ever used bioinformatics software on Bridges2 (you should it's a great resource), you have used biocontainers.

Summary

If you're new to Docker, think of it as software tupperware. Instead of asking you to install 20+ different bioinformatics tools (each with their own dependencies, version requirements, and quirks), Docker bundles them all into a single portable image. I also have it setting up the networking so the tools are visible via an internet browser. Docker can shrink an amazing number of processes into a single step.

Benefits:

- **Consistency:** Everyone using this platform, on any operating system that will run docker, will run the exact same versions of all tools.
- **Reproducibility:** You won't have to worry that your local R package versions are slightly different than mine.
- **Isolation:** The tools live inside the container. They won't mess with your global environment.

2. Clone the Repository and Build Container

First things first: grab the code from GitHub.

Open a terminal on your Jetstream2 VM and run:

```
git clone https://github.com/kallistaconsulting/genomic_resources_clogmia.git
cd genomic_resources_clogmia
```

Use the ls command to look into this folder and confirm that you see the same files that are listed at the top of the github repository. At very least, you should see setup.sh and dockerfile. Confirming you see these, then run

```
sudo bash setup.sh
```

This runs a wrapper script that handles the docker build and run commands. Within these commands, the dockerfile is used to build the contents inside a container, then run the container with the correct mapping of network addresses for the various tools. This command also captures your VM-assigned IP

address and includes it in the container. This line makes it so JBrowse2 and SequenceServer2.0 work on every new VM with randomly assigned IPs. (IPs are just the numeric webaddresses you see when troubleshooting your router – 10.0.0.1, 197.0.1.1, etc).

Note: This setup command will take about 30 minutes to install, mainly due to installation of R packages. Testing indicated 40Gb root directory was sufficient for the Jetstream VM set up, 20Gb was not.

From here, you can run any of the tools from a web browser with the proper links:

IP:3000?config=config.json → JBrowse2 instance

IP:3838/freeCount/apps/DA/ → edgeR differential expression applications

IP:3838/freeCount/apps/FA/ → topGO and GSEA functional enrichment applications

IP:3838/crisprFinder → views pre-profiled transcriptome to aid in initial design of sgRNA

IP:3838/crisprViewer → views pre-created crispr sgRNA primer design tables

IP:3838/crispRdesignR → views default crispr primer design tool with preloaded references

IP:4567 → SequenceServer2.0 instance with blast databases and JBrowse2 link backs

3. Build the Website (Optional, but recommended)

The tools for this platform are integrated into a pre-configured Drupal website that is customizable as any Drupal site, but organizes the tools and resources in point and click interface to make it easy to remember. There are additional items in the website such as pages for downloading the references used, publications associated with the site, etc. An example is currently available at: http://149.165.151.125/home

This step takes ~5m to set up.

Installing the website

1. On the local machine where you've installed the tools (step 2):

```
cd genomic_resources_clogmia/drupal
```

2. Edit the init.sql file. This file defines your username, database name, and password. You will want to edit this from the defaults to provide security, as the defaults are used and they are publicly known. Change 'drupalpass' to any password you would like to use. Use this password for all following passwords mentioned.

3. Run the install script:

```
sudo bash setup.sh
```

While installing, it will also ask you a couple of other questions. Testing was done with:

there is no root password, hit enter

unix_socket auth Y

reset root password (for ease, I set it the same as step 2)

remove anon users Y

disallow root login remotely

remove test database Y

reload privilege tables now Y.

You will then be asked to enter a password twice, use the one you set above.

4. You should now be able to access the website from your own IP (e.g. as you could with 149.165.173.182). You will see a Drupal setup page, which you only have to do once. I tested this installation with Standard installation, and the information from the init.sql file (drupal is your database by default, drupaluser as username, and whatever you set as root password). Save and continue, reload your browser. The site should come up automatically! Tools are listed in the menu and walked through in the following chapters.

4. Secure the Website

For security, please IMMEDIATELY:

1. Go into the Access tab and log in as user: admin, password: clogmia

2. Now the drupal admin menu will appear at the top of the site. Click people, and next to admin, click edit.

3. Change your password by typing the preset password (clogmia) at the top under 'Current password', then your new password next to 'Password' and again below when prompted. Scroll to the bottom and save. You now have secured access to this drupal site and can customize with basic Drupal methods.

5. Troubleshooting & Common Issues

Even though the install is designed to be smooth, here are some issues you might encounter and how to fix them (added to as they come up):

Port Conflicts

- **Symptom:** Error like `bind: address already in use` when trying to run `docker run`.
- **Cause:** Something else is already listening on port 80. If you are running multiple browsers, this will also happen as they are all trying to map the the same port. You need to assign new ports or remove the overlap.
- **Fix:** Use a different port mapping, e.g. running multiple browsers:

    - edit the setup.sh file in the main repository. Where it has “-p 3838:3838” change this to something like “-p 3839:3838”. The second number cannot change, that is the value inside the container. However, you can connect that to any port on the outside – I tend to keep them in order. You will also need to name the container something else.
    - A second browser may have a setup file that looks like:

```
docker build -t genome_browser .
docker run -d \
  --name genome_browser \
  -e HOST_IP=$(curl -s ifconfig.me) \
  -p 3839:3838 \
  -p 3001:3000 \
  -p 4568:4567 \
  genome_browser2
```

- **Fix:** You have an old container you need to turn off.
    - Use the command: “docker container stop genome_browser”, then “docker container rm genome_browser”

Out of Memory Error

- **Symptom:** Build fails with errors during package installation (especially R or BLAST).
- **Cause:** Jetstream2 VM doesn’t have enough RAM allocated.
- **Fix:** Use a larger flavor (e.g., `m3.medium` or higher). For genomics workloads, 16–32 GB RAM is a safe bet.

6. Wrapping Up

You’ve just installed the entire Genomic Resources Browser ecosystem on a fresh Jetstream2 VM. That’s no small feat—you now have:

- An upgraded JBrowse2 running in your browser.
- BLAST, crisprDesignr, and all the backend tools already configured.
- A working test dataset to confirm everything is functional.

From here, you’re ready to dive into the real fun: exploring your RNA-seq data, searching genomes, and building custom CRISPR guides.

In the next tutorial (Case Study 1), we’ll jump straight into visualizing RNA-seq tracks in JBrowse2.

Case Study 1: Viewing RNA-seq Data in JBrowse2

Outline:

1. Introduction and Input Data
2. Loading data into the browser
3. Navigating JBrowse2
4. Unique features of this platform
5. Wrapping up

1. Introduction and Input Data

You have received new RNAseq data and would like to view it against the genome, gene models, and developmental expression patterns. It's a common use for genome browsers, and a useful way to explore your data.

The data provided and the demo as written can run on a Jetstream2 VM as described above in the initial set up. This is not ideal, but it can be done. Real data should not be processed on a VM. ACCESS, which provides access to Jetstream2, also provides access to more suitable machines, such as Bridges2. Bridges2 has most, if not all, software installed, has many more CPU and memory resources, and a job scheduler to make it much more convenient. Quality control (QC), formatting of your data, even mapping your data to the genome will go smoother and faster on Bridges2 than on the VM. Best practice is to run the processes on a computer with more resources, then transfer the final files over to the VM with the genome browser on it.

With any new data, you will need to first QC your raw read data (resources here) then map the cleaned data to the genome, producing a .bam file and a .bai index file. In this demo, we will skip the QC step, but will map the reads to the genome and index the files. We will use published RNAseq data from the publication (SRR31734262). This is RNAseq data from a wild-type *Clogmia albipunctata* embryo, which we will use the first 2000 reads as a demo.

2. Loading data into the browser

Start with your JBrowse2 instance is running (if you're not there yet, head back to **Case Study 0: Installation Walkthrough**).

Data acquisition and prep (skip if you have a BAM file already). This is a step better run on a machine with more resources and space, but the demo can be done on the VM if you follow the steps below to clean up as you go.

i. sign into your VM.

```
#If on Jetstream2, replace 0.0.0.0 with your IP:

ssh exouser@0.0.0.0
```

ii. install samtools, hisat2, sratoolkit

```
############ Install directory creation ############

#We will install all software into ~/local/.

cd ~

mkdir local


############ Samtools Installation ############

cd ~/local

wget https://github.com/samtools/samtools/releases/download/1.22.1/samtools-
1.22.1.tar.bz2

tar xfv samtools-1.22.1.tar.bz2

rm samtools-1.22.1.tar.bz2

cd samtools-1.22.1/

./configure --prefix=$HOME/local/

make

make install


############ HiSat2 Installation ############

cd ~/local

wget https://cloud.biohpc.swmed.edu/index.php/s/fE9QCsX3NH4QwBi/download/hisat2-2.2.1-
source.zip

unzip hisat2-2.2.1-source.zip

rm hisat2-2.2.1-source.zip

cd hisat2-2.2.1/

make

cp * ~/local/bin


############ SRA-Tools Installation ############

cd ~/local

wget https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/3.2.1/sratoolkit.3.2.1-ubuntu64.tar.gz

tar xfv sratoolkit.3.2.1-ubuntu64.tar.gz

rm sratoolkit.3.2.1-ubuntu64.tar.gz


############ Set Path ############

#Set path to install location
```

```
PATH=$PATH:$HOME/local/bin
```

iii. Clone a premade pipeline for mapping (More details from NCGAS)

```
cd ~

git clone https://github.com/kallistaconsulting/MDIBL_pipeline_for_counts.git
```

iv. Pull the genome and gff from the website

```
############ Grab Genome and GFF from Github ############

cd ~/MDIBL_pipeline_for_counts/Reference

rm *dpul* && rm splicesites.txt #removes demo files for pipeline

wget https://github.com/kallistaconsulting/genomic_resources_clogmia/releases/download/
v1.0.3/genome-resources-clogmia.tar.gz

tar xfv genome-resources-clogmia.tar.gz --strip-components=1
genome-resources-clogmia/genome_files/Clogmia_genome_vNCBI.fa

tar xfv genome-resources-clogmia.tar.gz --strip-components=1
genome-resources-clogmia/genome_files/Clogmia_vNCBI.gff3

rm -R genome-resources-clogmia.tar.gz #frees up a lot of space!!
```

v. Build index for genome using htseq2-build

```
cd ~/MDIBL_pipeline_for_counts/Reference

python3 ~/local/hisat2-2.2.1/hisat2-build Clogmia_genome_vNCBI.fa Clogmia
```

vi. grab RNAseq data from NCBI

```
cd ~/MDIBL_pipeline_for_counts/Raw_Sequences

rm * #removes the demo files for this pipeline

~/local/sratoolkit.3.2.1-ubuntu64/bin/fastq-dump --split-files -X 2000 SRR31734262
```

vii. Map the reads to the genome using MDIBL pipeline. This is just an automation of the standard hisat2 workflow.

```
cd ../Scripts

nano 1_hisat_and_samtools.job.blank
```

Edit the file to match:

```
----------------

ref=../Reference/ Clogmia


stub=


reads="../Raw_Sequences/"

left="_1.fastq"

right="_2.fastq"
```

```
sam=../Mapping/SamFiles/$stub.sam

summary=../Mapping/SummaryFiles/$stub.hisat2.summary


#uncomment for paired reads

hisat2 -p 2 -x $ref -1 $reads$stub$left -2 $reads$stub$right -S $sam --max-intronlen 5000 --new-summary --summary-file $summary

#uncomment for single reads

#hisat2 -p 2 -x $ref -U $reads$stub$left -S $sam --max-intronlen 5000 --new-summary --summary-file $summary


samtools sort $sam > ${sam%sam}sort.bam

samtools index ${sam%sam}sort.bam

-----------------
```

viii. Exit the file using ctrl+o to write out, ctrl+x to exit. Then, run the setup for this pipeline

```
bash make_list.sh          #populate names from Raw_Sequences directory

bash make_1.listed.sh      #make a file for each sequence in Raw_Sequences directory


./SRR31734262.hisat_and_sam.job

#wait as this processes


#clean up is important on a VM

cd ../Mapping/SamFiles

rm SRR31734262.sam  #sam is the same as bam but ~6x larger iirc. Redundant, remove
```

ix. Download your bam and index, remove all temporary files to save space on your VM.

```
############ Pull Files ############

#On a fresh terminal signed into your local computer

#Replace 0.0.0.0 with your IP


cd ~/Downloads  #or wherever you want to save the files

scp exouser@0.0.0.0:~/MDIBL_pipeline_for_counts/Mapping/SamFiles/*bam* .


#confirm files transferred with checksum, comparing size, etc


############ Clean Up ############

#back on your VM terminal:

cd      #takes you home
```

```
rm -Rf MDIBL_pipeline_for_counts
#remove all files, you only need the bam and bai you transferred
```

You now have a bam and a bai file to upload to JBrowse2!

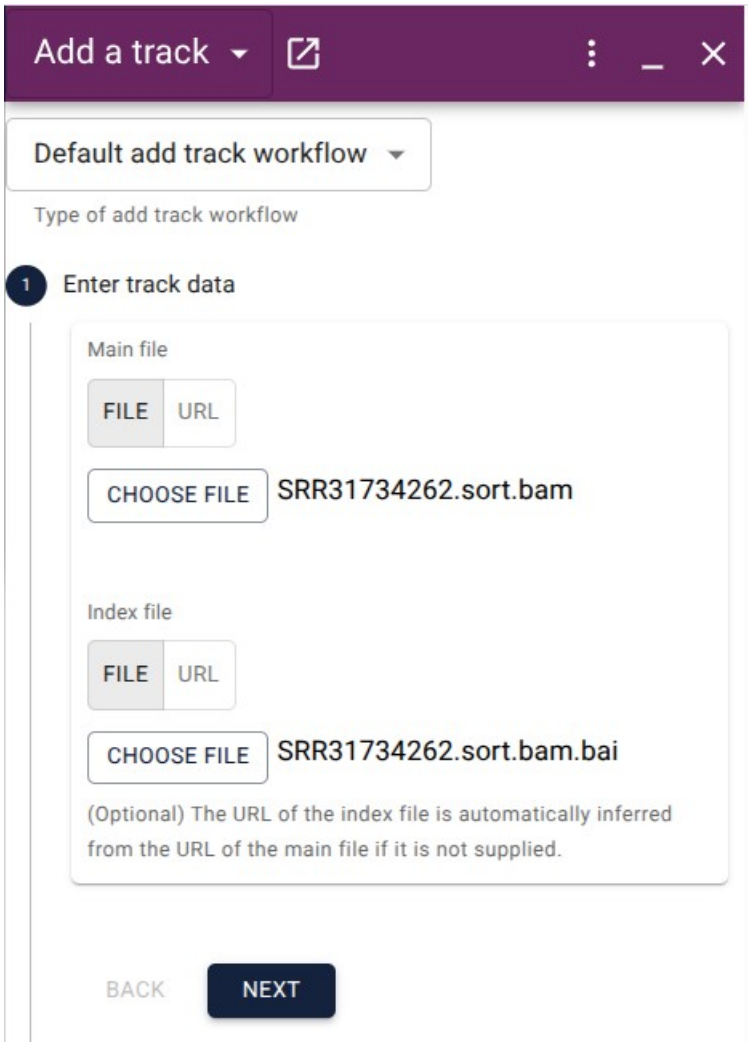
3. Load a BAM File

i. In your browser, navigate to your running instance (e.g., `http://<your-VM-IP>:3838` `or use the demo site`).

ii. Click into the **JBrowse2** view. You'll see the reference genome and annotation tracks already loaded.In the JBrowse2 menu, click the **"File"** button.

iii. Choose **"Open track"**.

iv. Under Main file, click File, then select your BAM file (e.g., `SRR31763336.sort.bam`).



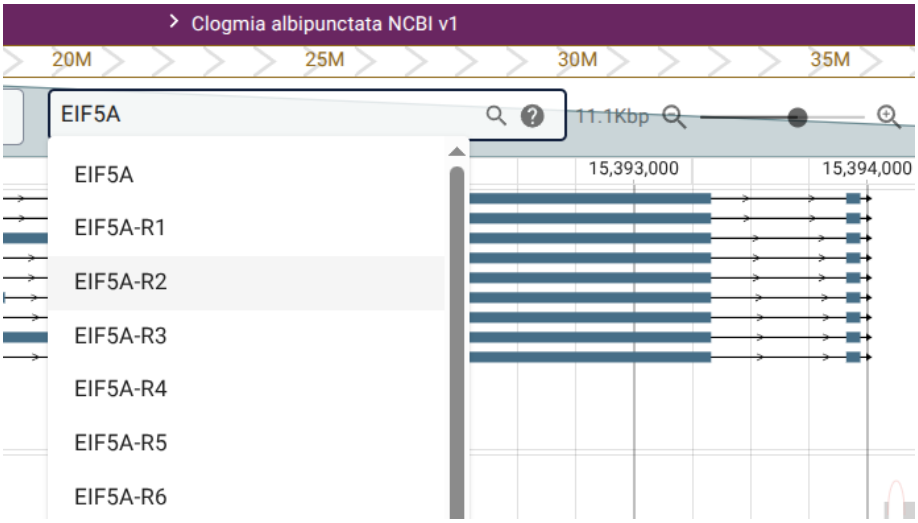
v. Under Index file, click File, then select your BAI file (e.g., `SRR31763336.sort.bam.bai`).

vi. Click Next.

vii. You can rename the track here if you wish, but I will leave it as is. Leave the track as BAM adapter (as you loaded a BAM file!), leave the adapter type as Alignments track, and leave the Assembly as is.

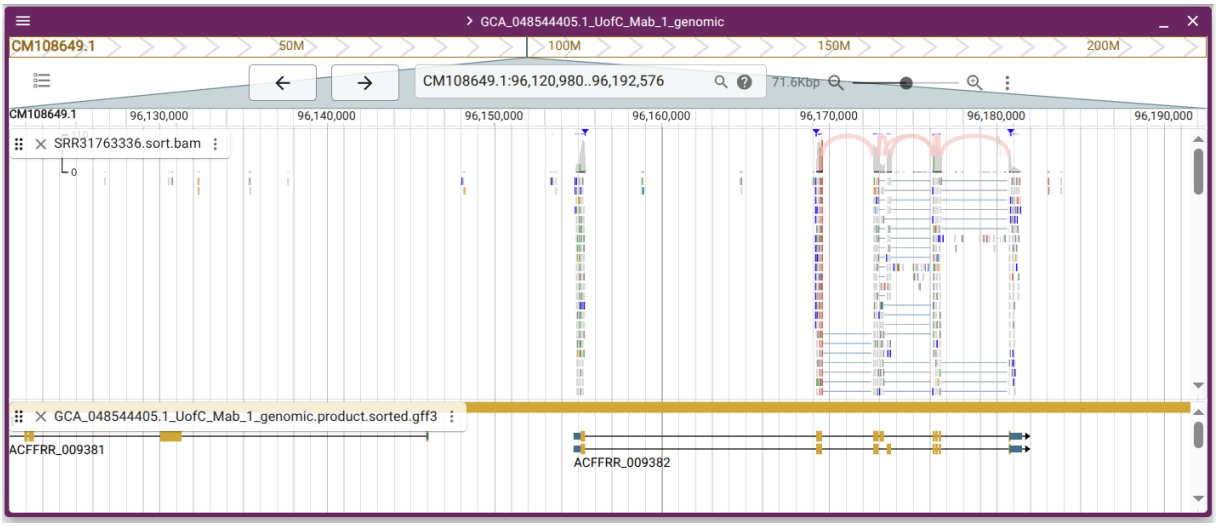
viii. Click ADD.

ix. Since we only loaded some of the reads, you can find a region with data by searching for EIF5A by typing it in the location textbox



x. If you load a full file, you will have to zoom in to see the files. Zoom by clicking the magnifying glass or by highlighting a section of the location bar (with numbers like 100M, 120M, etc). Keep zooming in until you see the track load.

- On the top portion of the track, you can see the histograph of reads, with bridges across gaps. Remember, RNA is edited from the genomic template, so you will see gaps. Areas of high read density will appear as taller coverage regions; sparse or empty regions will look blank.
- If you zoom in close enough, you'll be able to see individual reads, including (below is an example from a different browser with more data loaded):
    - Mismatches vs. the reference (often colored bases).
    - Insertions/deletions (small gaps or bumps).



You can also view this data against the gene models as well. Click the “Clogmia_vNCBI.sorted.gff” track. This is the gff and holds the gene models. You are likely to see the coverage match up with where there are exons on the gene models.

4. Navigating JBrowse2 Basics

Zooming In and Out

At the top of the genome view, you’ll see a navigation bar with zoom controls:

- Zoom in: click the “+” button or scroll in with your mouse/trackpad.
- Zoom out: click the “–” button or scroll out.

What you should see:

- At a broad zoom level, you’ll see the entire chromosome scaffold with features shown as thin blocks.
- As you zoom in, gene models expand, introns/exons become visible, and aligned reads from your BAM file start to appear.
- Zoomed all the way in, you’ll see base-by-base alignments of your sequencing reads against the reference sequence.

NOTE: JBrowse2 loads data in chunks. When you zoom or scroll into a new region, give it a moment to fetch reads or annotations.

Uploading BAM Data (Review – walk through above)

If you didn’t already add your BAM file:

1. In the track menu, click “Add” → “Open track file or URL.”

2. Select your BAM file and ensure its `.bai` index is in the same folder.

Once loaded, you'll see your alignments drawn directly under the genome track.

Viewing BAM Data

Once your BAM is loaded (see Section 2), here's how to interpret it:

- **Coverage Track:** Shows read depth across the locus. Peaks indicate highly expressed exons; valleys may suggest low expression or splicing.
- **Individual Reads:** When zoomed in, you'll see colored rectangles for each aligned read.
    - **Mismatches**: Bases that differ from the reference are often highlighted.
    - **Splice junctions**: Reads may appear "split," with arcs skipping across introns — very common in RNA-seq data.

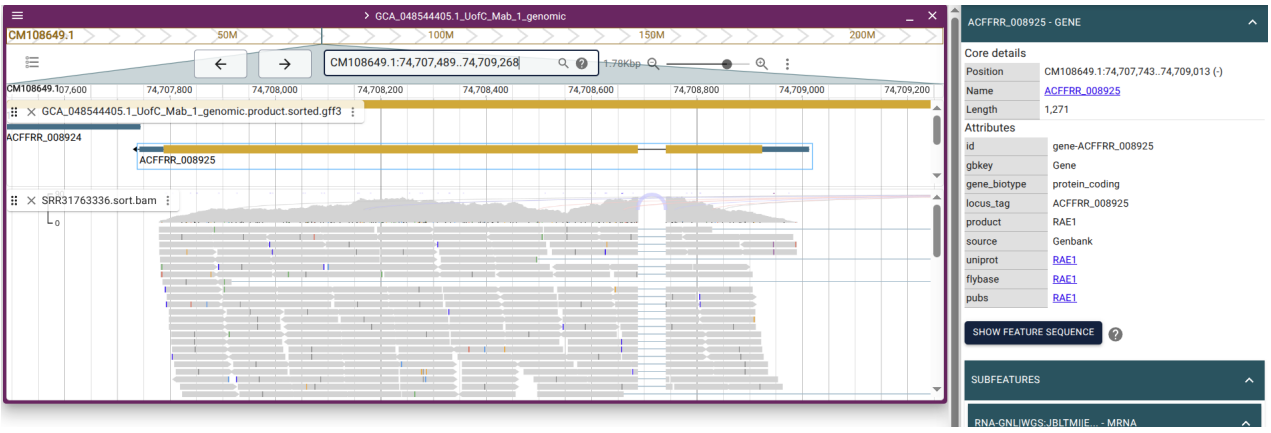
What you should see for `EIF5A`:

- Higher coverage across exons in the histogram bar (top line with arches and stacks).
- Splice junctions represented as split reads bridging across an intron.

More on Searching for Specific Loci

The search box at the top is your entry point for exploring specific genes or coordinates. In the search box, type: `OPA1`

- You should see the browser jump to the region containing this gene.
- The gene model track should highlight `EIF5A`, with exons/introns laid out along the reference.
- If you have your full RNA-seq BAM file loaded, you'll also see coverage piling up over exons, confirming transcription. (below is an example from another browser with more data loaded).



These searches let you jump directly to features of interest instead of endlessly scrolling through scaffolds. You can also see that there are links for RAE1 in the gene card. This is the gene product for this gene (not available for all loci). Clicking on any one of these will link you to more context.

Data Options in JBrowse2

JBrowse2 supports a wide range of file formats beyond BAM and BigWig: VCF (variants), GFF/GTF (annotations), CRAM, etc. Each format adds another perspective on the genome.

To explore these possibilities without touching your own data yet, you can play around with the official **Volvox demo site**: JBrowse2 Demo: Volvox Genome

This demo includes:

- Gene annotations.
- Example alignments.
- Expression coverage.
- Variants and synteny.

Other demos can be found here: https://jbrowse.org/jb2/demos/

5. Unique features of this platform

Database link outs

If you click on a gene model, you will see a gene card that pop up, which we will use more in the next demo. At the bottom of the first section, you can see that there are links for uniprot, flybase, and google scholar to help provide context for any gene of interest. Further links will be added through time.

6. Wrapping Up

- JBrowse2 helps provide a way to view multiple data types against each other.
- JBrowse2 is anchored primarily to locations, so all data must be mapped to the genome.
- While we walked through RNAseq data here, there are numerous other types of data that can be uploaded in the same way.

7. Troubleshooting

Remember if you run out of space, most cloud platforms allow you to add volumes to the VM. This acts as a kind of USB stick that you can temporarily add to increase working space. Alternatively, if you know you are going to be doing a lot of analysis or track loading, you can simply create a larger VM from the outset – making it 70Gb instead of 40Gb for instance.

Case Study 2: BLAST Search Within the Genome (and Beyond)

Outline:

1. Introduction and Input Data
2. Using SequenceServer2.0
3. Visualizing results
4. Cross-tool integration
5. Wrapping Up

1. Introduction and Input Data

Now that we have our genome and annotation and maybe even some reads loaded, you want to find the homologous transcript for a gene of interest found in Drosophila (OPA1). OPA was show to be of interest in the genome paper associated with this project. We can use the Blast tools on the platform to search the genome and reference back to the gene.

BLAST (Basic Local Alignment Search Tool) lets us take a sequence (nucleotide or protein) and search it against our genome to find matches. If duplicates (paralogs, gene families, pseudogenes) exist, BLAST will tell us where and how similar they are.

In this case study, we'll:

- Pull a gene sequence from flybase.
- Use SequenceServer 2.0 — a modern web front-end for BLAST that's included in the Browser platform.
- Run a search with a target gene sequence.
- Visualize results and explore how they connect back into JBrowse2.

Input data you'll need:

- The nucleotide sequence of your gene of interest (FASTA format). Let's grab it from Flybase. You can search for it manually and grab the decorated view, or go to https://flybase.org/decoratedfasta/FBgn0261276.
- You can also pull sequences from the browser if you are looking for duplicates within the genome. Open JBrowse and search for OPA1. This will pull up the location of the gene quickly.

- On the gene card, click SHOW FEATURE SEQUENCE

- Copy the sequence.

2. Using SequenceServer 2.0

i. Open SequenceServer

- Go back to your website, and click Tools, BLAST+, then the Blast via SequenceServer2.0 link. Or if you didn't install the website, you can go directly to your ip:4567 (0.0.0.0:4567 if your IP was 0.0.0.0).

ii. Paste Your Sequence

• Copy the DNA sequence of your target gene into the query box.

• SequenceServer will auto-detect whether it's nucleotide or protein.

iii. Choose a Database

• We want to search the transcripts, so select the Clogmia.transcripts file under Nucelotide databases.

iv. Pick Your BLAST Settings

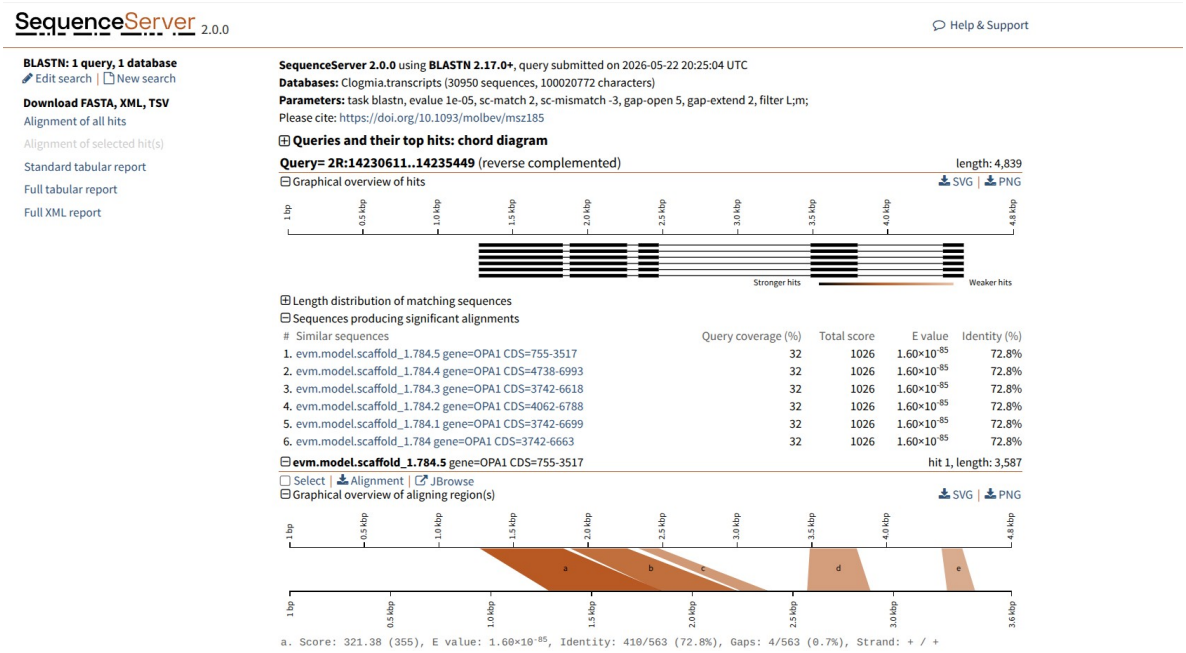
• You can ignore the advanced parameters for now, but know that you can tune blast to run however you normally set your cutoffs, simply by putting the options here. If you need a quick reference to the options, clicking the ? will list them for you.

v. Run the Search

• Hit the BLASTN button.

3. Visualizing Results

SequenceServer 2.0 presents results in a very user-friendly way compared to classic BLAST outut.



Some useful things to look at:

- Graphical overview will show you information on your numerous hits if you have them. Here we only hit one chromosome, so this is just showing one black line. Not terribly interesting.
- The individual hits will be visualized as well as given in standard alignment form for you to peruse. On the example, notice there is an a and a b score. This is because this query hit the same chromosome several times, but they look to all be different versions of the same location.
- You can download the reports you would normally receive from the left menu, under Download FASTA, XML, TSV.
- We have set up link outs to the JBrowse2 instance as well. This is a button available for each hit.

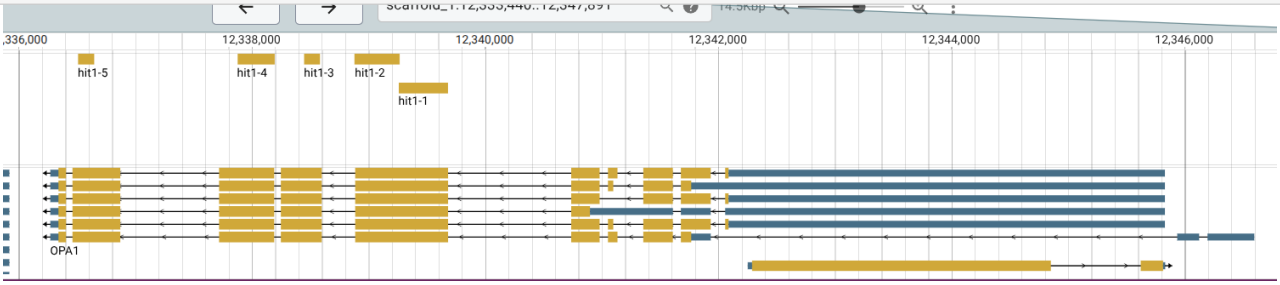
4. Cross-Tool Integration

Part of the convenience of this platform is that we have linked BLAST results and JBrowse2 to each other.

- Each hit in SequenceServer has a “JBrowse” button at the top.
- Clicking this takes you directly to JBrowse2 at the locus of that hit. You may want to zoom out slightly to get a better view:

- This links us directly to the transcript it hit.
- We can also blast against the genome, and it will show us our blast result against the genome. We can also pull up the annotation and manually see where they hits are in context of gene models or expression.

>2R:14230611..14235449 (reverse complemented)



- Here you will see the hits are still aligning to the same place, but because the exons are what are mapping to the genome, it is in fragments (introns were not solid hits across distant organisms). You will also see that they are in reverse order (hit 1-5, 1-4, 1-3, 1-2, 1-1) – this is because the sequence from flybase is the reverse complement – it's labeled ">2R:14230611..14235449 (reverse complemented)". However, there are a couple of exons not included in the flybase version that we do see in Clogmia!

5 . Customization Notes

How did we set this up, in case you want to edit it? Behind the scenes, SequenceServer uses a configuration file (links.rb) that tells it how to construct JBrowse links.

Location of links.rb:

- In the docker container: /var/lib/gems/3.0.0/gems/sequenceserver-2.0.0/lib/sequenceserver/links.rb
- In the github repository: genomic_resources_clogmia/sequenceserver/links.rb

6. Wrapping Up

By the end of this case study, you should be able to:

- Extract a gene sequence from JBrowse2.
- Run a BLAST search in SequenceServer 2.0
- Interpret tabular, graphical, and alignment results.
- Jump back into JBrowse2 from BLAST results to see hits in full genomic context.

This workflow is central for questions like:

- *Does my gene have paralogs?*
- *Where are gene family members located?*
- *Is this sequence unique enough for downstream assays (e.g., CRISPR guide design)?*

Case Study 3: Designing an sgRNA with crisprDesignr

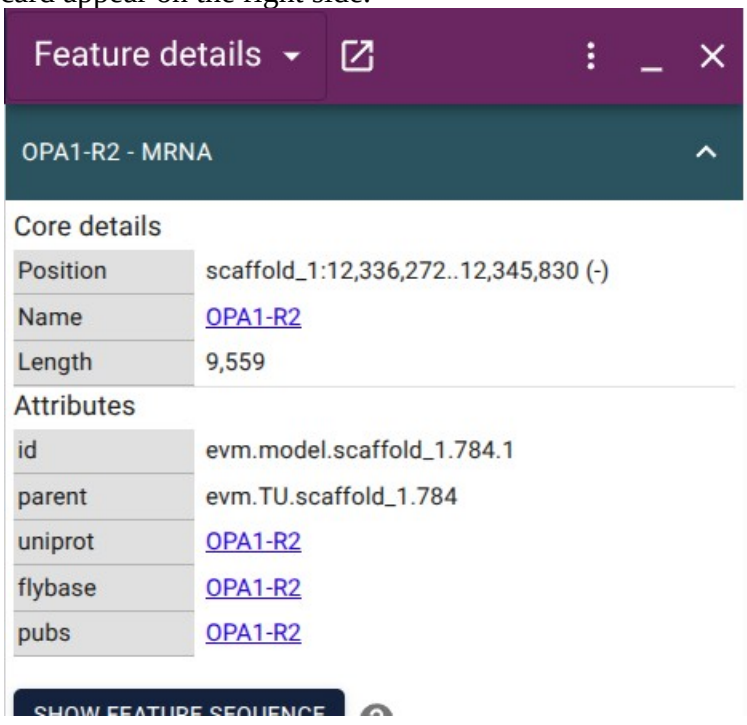
Now that we have identified a gene of interest, let's create an sgRNA primer for CRISPR Cas9 while considering off-targets, expression, and sequence context.

Outline:

7. Introduction and Input Data
8. Using CrisprFinder
9. Using CrispDesignr for unprofiled transcripts
10. Using CrisprViewer (coming soon)
11. Cross-tool integration
12. Wrapping Up

1. Introduction and Input Data

Pull up your JBrowse2 instance and search for the OPA1 gene again. Click on any of the transcripts and you should see a gene card appear on the right side.

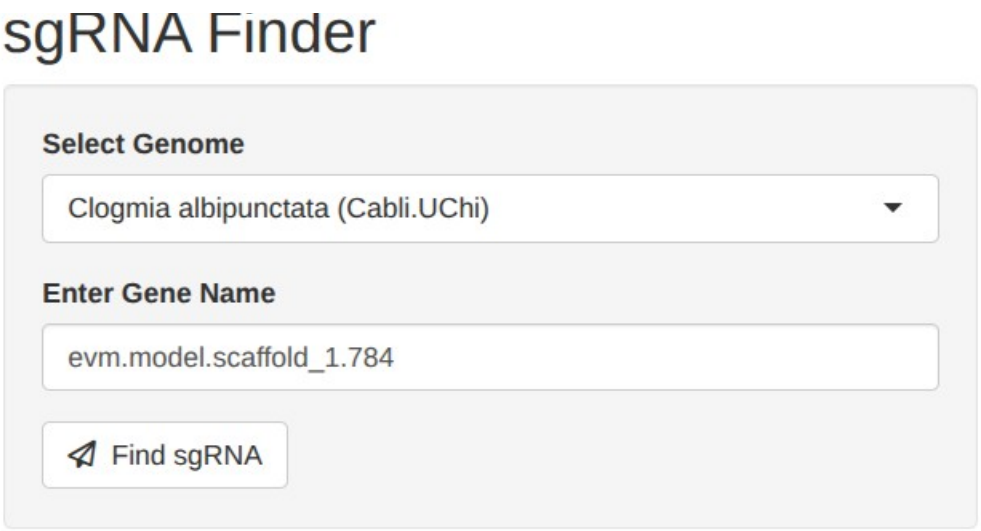


Here we want to grab the id, but only the information before the last ".". This is the transcript id, the .1 means this was the first transcript version identified. Here we want "evm.model.scaffold_1.784". We can use this to pull up the data from the pre-run crispr profiling included in the site.

2. Using CrisprDesignr

Go back to your main website and click Tools in the top menu, then CRISPR, then click the link to CrisprFinder. CrisprFinder is an edited version of the CrisprDesignr suite that allows you to view pre-profiled transcripts.

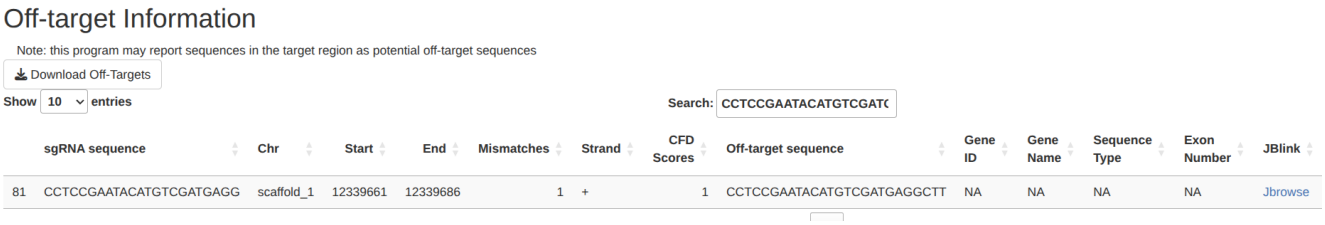
Once this loads, you will see that the genome is pre-loaded (as a BSgenome object), and you simply have to enter your transcript. Not all transcripts are profiled yet, more on that in a minute. For now, enter the information we grabbed from the browser and click Find sgRNA.



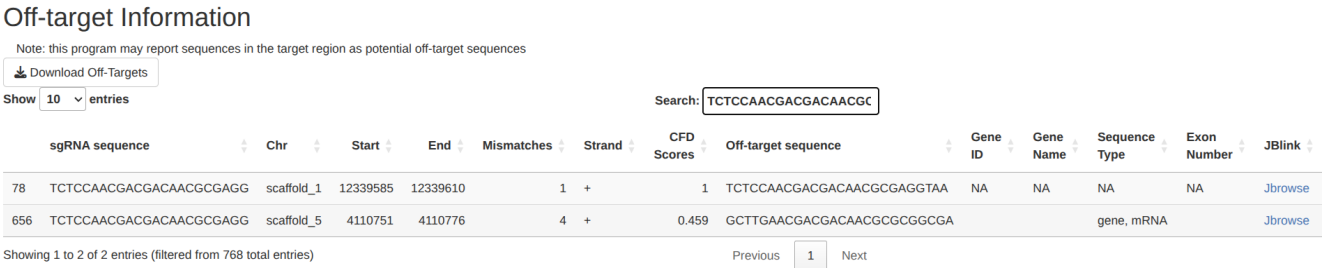
This will pull up tables with information for each PAM found for this transcript, as well as some information to help you identify the best candidates. GC content is listed, which you want between 30-80%, where it falls in the gene, if it is a homopolymer (not ideal), if it is self-complementary (not ideal), it's efficiency score (based on Doench et al 2016), and how many mismatches hit other locations. There is a Notes Code section that simply lists the problems with each sgRNA sequence. All of these columns are sortable, so you can prioritize your own needs. Further details can be found in the crispRdesignR publication (Beeber and Chain 2019).

Our transcript on interest has 246 possible sgRNAs that could be made using NGG PAMs. The 4th listed looks good. Let's copy the sequence, and use the next table, which will tell us if that sgRNA has off-target hits in the genome, which could be problematic.

You can put this sequence in the Search textbox above the second table. This will reduce the table to only the off target hits for that sequence.



Here there is only one off target listed, and it does not appear to be linked to any gene-coding region, which is a good sign. If we look at the 7th listed sgRNA option, we can see that this is not the case:



We can get more information on the off-target gene by clicking on the Jbrowse link, which will show you the off-target location (COMING SOON). This allows you to not only get more information about the off-target loci, but also view this against your RNAseq data to see if there is expression of this transcript at your targeted use. For instance, if they transcript is heavily expressed in a developmental stage, it would not be ideal to create a knockout using this sgRNA, as the modified zygote would likely fail to develop.

3. Using crispRdesignR for unprofiled transcripts

Not all genes are pre-profiled yet, as they take a bit of time to process. If you would like to, you can use the base crisprDesignr package locally or on your VM to profile any gene you are interested in by inputting it's sequence. This can take 15m to several hours.

Please visit https://github.com/dylanbeeber/crispRdesignR for detailed instructions on all the functions of this package.

i. Run crispRdesignR locally in R

To analyze a novel sequence, either outside of a transcript body or one that we have not profiled yet, you will first need the BSgenome object, which is available within the github release files. On any linux system, you can do the following:

```
#####Pull BSGenome Object from release############################

wget https://github.com/kallistaconsulting/genomic_resources_clogmia/releases/
download/v1.0.5/genome-resources-clogmia.tar.gz

tar xfv genome-resources-clogmia.tar.gz –strip-components=1 genome-resources-
clogmia/apps/crisprFinder/Bsgenome.Calbi.Uchi_1.0.0.tar.gz
```

or simply download the release and pull the file from that location. You will need to install it in R using:

```
R CMD INSTALL /location/Bsgenome.Calbi.Uchi_1.0.0.tar.gz
```

You will also need the gff, which you can pull from the Downloads page of the website that ships with the VM, or you can pull it from the resource release with the following command:

```
#####Pull GFF Object from release############################

wget https://github.com/kallistaconsulting/genomic_resources_clogmia/releases/
download/v1.0.5/genome-resources-clogmia.tar.gz

tar xfv genome-resources-clogmia.tar.gz –strip-components=1 genome-resources-
clogmia/genome_files/Clogmia_vNCBI.gff3
```

Then, open R. You will need Bioconductor installed, and the install crispRdesignR:

```
install.packages(“crispRdesignR”)
```

Finally, to boot the interface locally, load the package and call up the UI:

```
library(crispRdesignR)
crispRdesignRUI()
```

ii. Run crispRdesignR on the VM

A version of crispRdesignR is available on the VM. You can access it by opening a browser and navigating to YOUR_IP:3838/crispRdesignR.

This version will look slightly different than the one above. It does not require you to load the BSGenome object, nor does it require you to load the gff. These are hard coded into the app, as they are in a standard location on the VM.

Please note that this shiny app can take a long time to profile long sequences. Please check to see if the target is pre-profiled using CrisprFinder or run it locally if the VM lacks the resources to complete this task.

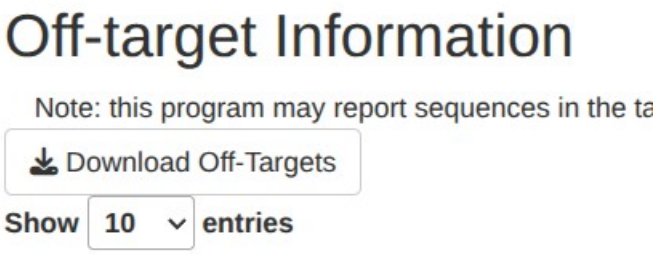
4. Using CrisprViewer

If you ran the default version of crispRdesignR locally, or you have data that you saved from prior analyses, you can view this output on the site or share it with collaborators. For this demo, we’ll use the OPA1 tables that were already pre-run.

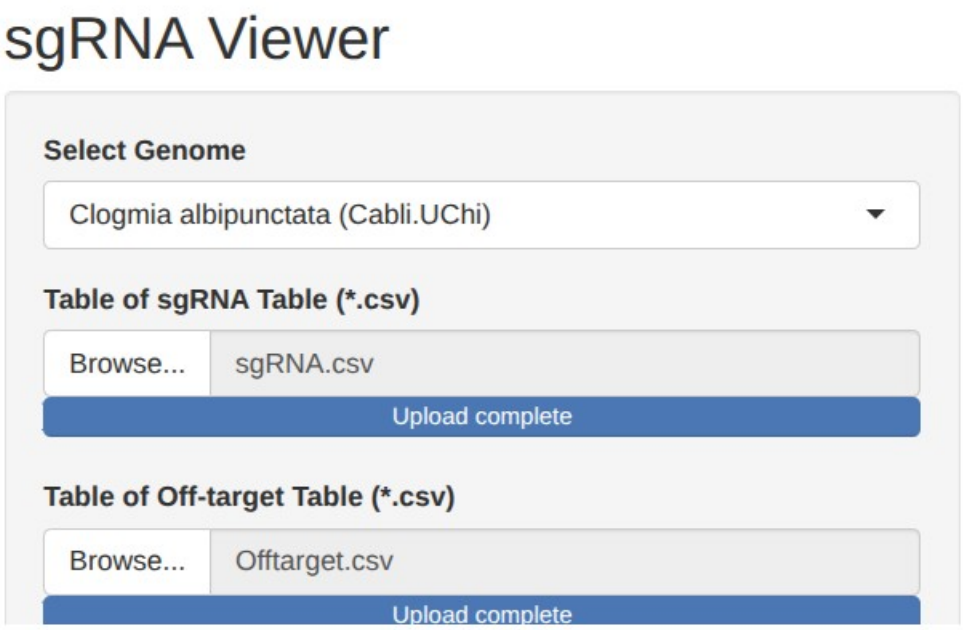
Pull up OPA1 in crisprFinder again, using the gene id as before.

Click the Download sgRNA button under the sgRNA Table header to download the first table.

Click the Download Off-Targets button under the Off-target Information header to download the second table.



Go to the website on the VM and click Tools, CRISPR, and then click the crisprViewer link. Here, you will see the same BSgenome Object preselected, and a location to upload your tables.



Click Find sgRNA to load the tables. This also populates the link to JBrowse that will be missing if you profile a sequence locally. This is designed to allow you to profile novel sequences using larger hardware while still benefiting from the ease of visualization.

5. Cross-tool Integration

The main integration here are

- the ability to view the off-target sites in Jbrowse, allowing you to view more information on off-target hits, and compare your ATACseq, RNAseq, or other data against that loci for even more context.
- access the pre-profiled transcripts. Each transcript can take ~15m to run manually, and it can be arduous to manually search each off-target transcript.
- Handling for viewing previously run tables with links auto populated, increasing your ability to share results with collaborators.

This platform seeks to streamline the process by reducing time to review profiles as well as evaluate their impacts.

6. Wrapping Up

By the end of this case study, you should be able to:

- Access pre-profiled transcripts for sgRNA candidates
- Evaluate the sgRNA candidates via different metrics (GC content, homopolymerization, etc)
- Search and view off-target hits in JBrowse.
- Profile your own transcripts as needed.