VERSION 2

JAN 23, 2024

ICGRC Portal Tripal Data Generation and Setup V.2

Locedie Mansueto[1], ramil.mauleon[1], Tobias Kretzschmar[1], Graham King[1]

[1]Southern Cross University

Locedie Mansueto

ABSTRACT

The data provided by the International Cannabis Genomics Research Consortium (ICGRC, https://icgrc.info) web portal consists of published results from past analyses, and results generated using the steps described in this protocol. We used public datasets, and open software commonly used for the specific tasks, suggested from best practices, or from other Tripal sites. The sections in these protocol can be grouped into Data Generation (steps 1-8), Setup Tripal and Modules (steps 9-15), Multi-omics API Integration (step 16,17), and Setup of non-Tripal pages (step 18).

External link:
http://www.icgrc.info

Protocol Citation: Locedie Mansueto, ramil.mauleon, Tobias Kretzschmar, Graham King 2024. ICGRC Portal Tripal Data Generation and Setup. **protocols.io** https://protocols.io/view/icgrc-portal-tripal-data-generation-and-setup-c27nyhmeVersion created by Locedie Mansueto

Protocol status: Working
We use this protocol and it's working

Created: Oct 09, 2023

Last Modified: Jan 23, 2024

PROTOCOL integer ID: 89038

Keywords: Tripal, Cannabis, Database, Omics

Prepare RNA-Seq Sequences

1 Next Generation Sequencing (NGS) RNA-Seq sequences are downloaded from NCBI SRA, and trimmed to remove adapters. Three sets of RNA-Seqs are prepared for: 1) Purple Kush gene prediction, 2) Finola gene prediction, 3) transcript assembly, expression level, and variant discovery for trichomes from 21 samples

1.1 Download and extract sequences from NCBI-SRA. The SRR fastq sequences can be paired-end (in SRRLIST_PE) or single end (in SRRLIST_SE)

purple_kush_srr_SE.txt 0B | purple_kush_srr_PE.txt 0B | finola_srr_PE.txt 0B | finola_srr_SE.txt 0B | 21trichs_srr_PE.txt 1KB

Software	
sratoolkit	NAME

Command

```
SRATOOLKIT_DIR=/path/to/sratoolkit
FASTQDOWNLOAD_DIR=
SRRLIST_PE=
SRRLIST_SE=
cat $SRRLIST_SE $SRRLIST_PE > $SRRLIST
cat  $SRRLIST | ${SRATOOLKIT_DIR}/fastq-dump --split-files --gzip --
outdir $FASTQDOWNLOAD_DIR -
```

1.2 Trim using trimmomatic. The paired-end and single-end fastq are iterated separately since they require different command arguments

Software

Trimmomatic	NAME

Command

```
TRIMMOMATIC_DIR=/path/to/trimmomatic
FASTQTRIM_DIR=
THREADS=10
lines=$(cat $SRRLIST_PE)
for line in $lines
do
      java  -jar $TRIMMOMATIC_DIR/trimmomatic.jar PE -threads $THREADS
 $FASTQDOWNLOAD_DIR/${line}_1.fastq.gz
$FASTQDOWNLOAD_DIR/${line}_2.fastq.gz
$FASTQTRIM_DIR/${line}_1.fastq.gz
$FASTQTRIM_DIR/${line}_1.U.qtrim.fastq.gz
$FASTQTRIM_DIR/${line}_2.fastq.gz
$FASTQTRIM_DIR/${line}_2.U.qtrim.fastq.gz
ILLUMINACLIP:${TRIMMOMATIC_DIR}/adapters/TruSeq3-PE.fa:2:30:10
SLIDINGWINDOW:4:5 LEADING:5 TRAILING:5 MINLEN:25
done

lines=$(cat $SRRLIST_SE)
for line in $lines
do
      java  -jar $TRIMMOMATIC_DIR/trimmomatic.jar SE -threads $THREADS
 $FASTQDOWNLOAD_DIR/${line}_1.fastq.gz $FASTQTRIM_DIR/${line}.fastq.gz
ILLUMINACLIP:${TRIMMOMATIC_DIR}/adapters/TruSeq3-PE.fa:2:30:10
SLIDINGWINDOW:4:5 LEADING:5 TRAILING:5 MINLEN:25
done
```

1.3 (Optional) Concatenate multi-fastq samples. BioSample SAMN with multiple SRR fastq sequences

booth2020_srr.txt 0B

Command

```
lines=$(cat booth2020_srr.txt)
cd $FASTQTRIM_DIR
for line in $lines
do
        a=($(echo "$line" | tr '\t' '\n'))
        cat "${a[1]}_1.fastq.gz" "${a[2]}_1.fastq.gz"
"${a[3]}_1.fastq.gz" "${a[4]}_1.fastq.gz" > "${a[0]}_1.fastq.gz"
        cat "${a[1]}_2.fastq.gz" "${a[2]}_2.fastq.gz"
"${a[3]}_2.fastq.gz" "${a[4]}_2.fastq.gz" > "${a[0]}_2.fastq.gz"
done
cd ..
```

Gene models prediction

2 Generation of gff and fasta files. Unless gene models and annotation are available in RefSeq for a genome assembly like cs10, local gene prediction and annotation are preformed. Cultivar-specific RNA-Seq datasets are used whenever available.

Software

STAR NAME

Software

FINDER NAME

2.1 Generate metadata csv listing the RNA-Seq sequences to use. Examples for Purple Kush and Finola

finola_metadata.csv 1KB purple_kush_metadata.csv 0B

Prepare the STAR and olego genome index. FINDER wrongly parse a dot/period (.) in the contig IDs. As most contig IDs from NCBI ReqSeq/GeneBank have dots, rename the IDs by replacing dots with underscore

Command

```
METADATAFILE=purple_kush_metadata.csv
REF_FASTA=pkv5.fna

FINDER_WORKDIR=finder_pkv5
FINDER_OUTDIR=finder_out
FINDER=/path/to/finder
CPU=12

GENOME_FASTA=${REF_FASTA}.renamed.fna
GENOME_DIR_STAR=star_index_without_transcriptome
GENOME_DIR_OLEGO=olego_index
STAR=/path/to/star

cd $FINDER_WORKDIR
ln -s $FASTQTRIM_DIR trimfastq_rnaseq

sed 's/\./_/g' $REF_FASTA > $GENOME_FASTA
$STAR --runMode genomeGenerate --runThreadN $CPU \
   --genomeDir $GENOME_DIR_STAR --genomeSAindexNbases 12 \
   --genomeFastaFiles $GENOME_FASTA
../dep/olego/olegoindex -p $GENOME_DIR_OLEGO  --output_directory
$FINDER_OUTDIR
```

2.2 Run FINDER.

Command

```
$FINDER/finder --metadatafile $METADATAFILE \
   --output_directory $FINDER_OUTDIR --genome $GENOME \
   --cpu $CPU --genome_dir_star $GENOME_DIR_STAR \
   --genome_dir_olego $GENOME_DIR_OLEGO \
   --no_cleanup --preserve_raw_input_data
```

The final outputs are generated in the directory ${FINDER_WORKDIR}/${FINDER_OUTDIR}. It is observed that multiple gene models are overlapping and fragmented in a region. These models are clustered in the next step using the output gtf file.

2.3 Cluster gene models using gffread.
It is observed that multiple gene models are overlapping and fragmented in certain regions. These models are clustered in this step from the FINDER output gtf file.

Software

gffread NAME

Command

```
FINDER_GTF=${FINDER_WORKDIR}/${FINDER_OUTDIR}/final_GTF_files/combined_
with_CDS_high_conf.gtf
CLUSTERED_GFF=${FINDER_GTF}.clustered.gff
CLUSTERED_PEP=${FINDER_GTF}.clustered.pep.fasta
CLUSTERED_CDS=${FINDER_GTF}.clustered.cds.fasta
CLUSTERED_SPLICEDEXON=${FINDER_GTF}.clustered.exon.fasta

/path/to/gffread -g $GENOME_FASTA --merge -d ${FINDER_GTF}.dupinfo \
   -K -Q -Y -x $CLUSTERED_CDS -y $CLUSTERED_PEP \
   -w $CLUSTERED_SPLICEDEXON  -o $CLUSTERED_GFF $FINDER_OUTGTF
```

Merged gene models are listed in ${FINDER_GTF}.dupinfo. The clustered gene models in CLUSTERED_GFF is in gff format. The remaining files are sequences for CDS, exons and proteins in fasta format.

2.4 Revert the contig names in CLUSTERED_GFF back to the original names used in REF_FASTA. As an option, the clustered output gene model IDs (gene ids, cds id, protein ids, exon ids) may be renamed using your own nomenclature.

Gene functional annotation

3 Functional annotation by sequence homology using mmseqs2, is a fast sequence alignment alternative to BLAST.

Software

mmseqs2	NAME

Dataset

UniProt Plants NAME

https://www.uniprot.org/help/entries_since_rel_x LINK

3.1 Download the latest Uniprot viridiplantae proteins sequence and generate mmseqs2 database

Command

```
MMSEQDB_UNIPROT=sptr_plants
MMSEQ_DIR=/path/to/mmseqs
$MMSEQ_DIR/mmseqs createdb sptr_plants.fasta.gz $MMSEQDB_UNIPROT
```

3.2 Get best hit alignment between predicted proteins with Uniprot plants

Command

```
MM_RESULTDB=pkv5_sptr_plants_resultDB
MM_BESTRESULTDB=pkv5_sptr_plants_bestresultDB
$MMSEQ_DIR/mmseqs search $CLUSTERED_PEP ${MMSEQDB_UNIPROT} \
   $MM_RESULTDB tmp --start-sens 1 --sens-steps 3 -s 7
$MMSEQ_DIR/mmseqs filterdb  $MM_RESULTDB $MM_BESTRESULTDB  \
   --extract-lines 1
$MMSEQ_DIR/mmseqs convertalis $CLUSTERED_PEP \
   ${MMSEQDB_UNIPROT} $MM_BESTRESULTDB ${MM_BESTRESULTDB}.txt
```

Assign the homologs functional annotation and scores (target ID, e-value) to the Description and/or Note attribute in the GFF.

3.3 Functional annotation by Interpro. Submit CLUSTERED_PEP to InterproScan to any of the public Galaxy servers (EU,US or AU). Include Gene Onlogy ID (GO_ID) assignment, and set outputs to xml and tsv formats.

Transcript assembly and alignments

4 Four sources of Cannabis transcripts use are: 1) cs10 mRNA from NCBI RefSeq, 2) transcript assembled by publications, 3) gene prediction using cultivar-specific RNA-Seq sequences in step 2., 4) assembled transcripts using trichome RNA-Seq of 21 cultivars from multiple sources. The transcripts in this step are loaded and visualized in the JBrowse genome browser.

21trichomes_replicates.txt 1KB

4.1 (Optional) Assemble RNA-Seq sequences into transcripts. rnaspades is used for its speed.

Software	
SPADES	NAME

Command

```
SPADES_DIR=/path/to/spades
SPADES_OUTDIR=outdir-21trichomes

# concatenate fastq files
cat $FASTQTRIM_DIR/SRR*_1.fastq.gz > $FASTQTRIM_DIR/allsrr_1.fastq.gz
cat $FASTQTRIM_DIR/SRR*_2.fastq.gz > $FASTQTRIM_DIR/allsrr_2.fastq.gz


$SPADES_DIR/rnaspades.py  -o $SPADES_OUTDIR \
   --pe1-1 $FASTQTRIM_DIR/allsrr_1.fastq.gz --pe1-2
$FASTQTRIM_DIR/allsrr_2.fastq.gz \
   -t 10 -m 300

TRANSCRIPT_SPADES_FASTA=$SPADES_OUTDIR/hard_filtered_transcripts.fasta
```

The output assembled transcript sequences are in file $TRANSCRIPT_SPADES_FASTA

4.2 Align transcripts to genome into bam file. The results are visualized in the JBrowse genome browser.

Software

minimap2	NAME
Linux	OS
Heng Li	DEVELOPER
https://github.com/lh3/minimap2	SOURCE LINK

Software

samtools NAME

http://htslib.org/ DEVELOPER

GitHub SOURCE LINK

Command

```
MINIMAP2_DIR=/path/to/minimap2
SAMTOOLS_DIR=/path/to/samtools

TRANSCRIPT_FASTA=$TRANSCRIPT_SPADES_FASTA
TRANSCRIPT_BAM=${TRANSCRIPT_FASTA}.bam

$MINIMAP2_DIR/minimap2 -ax splice $REF_FASTA $TRANSCRIPT_FASTA |
$SAMTOOLS_DIR/samtools view -@ $THREADS -u -  | $SAMTOOLS_DIR/samtools
sort -m 4G -@ THREADS  -o $TRANSCRIPT_BAM  -
```

4.3 Find open reading frames (ORF) of assembled transcripts to within the reference genomes.
Transdecoder is used following https://github.com/TransDecoder/TransDecoder/wiki
The transcripts are aligned to genome using GMAP https://github.com/juliangehring/GMAP-GSNAP

Build the GMAP genome database

```
REF=/path/to/reference_genome.fasta
refgenome_name=cs10
TRANSCIPT_FASTA=$TRANSCRIPT_SPADES_FASTA
$GMAP_GFF=${refgenome_name }-${TRANSCIPT_FASTA}-gmap-f3n1.gff
gmap_build -D $refdb_name -d $refgenome_name -t 10 -s none  $REF
```

Map assembled transcripts to find ORFs

```
gmap -D $refdb_name -d $refgenome_name -f 3  -t 12 -n 1  >
$GMAP_GFF
```

Search for long ORFs and peptides from assembled transcripts

```
../../TransDecoder.LongOrfs -t $TRANSCIPT_FASTA
```

Longest ORF peptides are aligned to UniProt plants, using mmseqs similar to step 3.2. Best hit results is in file in LONGESTORF_MMSEQ_BESTHIT

```
mmseqs search rnablongestorf_pep_seqDB sptr_plants_2022_seqDB
rnablongestorf_pep-sptr_plants-resultDB tmp --start-sens 1 --sens-
steps 3 -s 7
mmseqs filterdb  rnablongestorf_pep-sptr_plants-resultDB
rnablongestorf_pep-sptr_plants-bestresultDB   --extract-lines 1
mmseqs convertalis rnablongestorf_pep_seqDB sptr_plants_2022_seqDB
rnablongestorf_pep-sptr_plants-bestresultDB
$LONGESTORF_MMSEQ_BESTHIT
```

Run TransDecoder ORF prediction, generating output ${TRANSCIPT_FASTA}.transdecoder.gff3

```
../../TransDecoder.Predict -t $TRANSCIPT_FASTA --retain_blastp_hits
$LONGESTORF_MMSEQ_BESTHIT
```

Generate ORF with genome coordinates

```
TRANSCRIPT_GENOME_GFF= ${TRANSCIPT_FASTA}.transdecoder.genome.gff3
perl ../../util/cdna_alignment_orf_to_genome_orf.pl
${TRANSCIPT_FASTA}.transdecoder.gff3 $GMAP_GFF  $TRANSCIPT_FASTA  >
$TRANSCRIPT_GENOME_GFF
```

Cluster overlapping gene loci, and generate CDS, exon and protein sequences

Software

gffread NAME

Pertea and Pertea DEVELOPER

https://github.com/gpertea/gffread SOURCE LINK

```
GENOME_FASTA=cs10.fna
FINDER_GTF=transcripts.fasta.renamed.fasta.transdecoder.genome.gff3
#
FINDER_WORKDIR}/${FINDER_OUTDIR}/final_GTF_files/combined_with_CDS_
high_conf.gtf
CLUSTERED_GFF=${TRANSCRIPT_GENOME_GFF}.clustered.gff
CLUSTERED_PEP=${TRANSCRIPT_GENOME_GFF}.clustered.pep.fasta
CLUSTERED_CDS=${TRANSCRIPT_GENOME_GFF}.clustered.cds.fasta
CLUSTERED_SPLICEDEXON=${TRANSCRIPT_GENOME_GFF}.clustered.exon.fasta

~/software/gffread/gffread -g $GENOME_FASTA --merge -d
${TRANSCRIPT_GENOME_GFF}.dupinfo -K -Q -Y -x $CLUSTERED_CDS -y
$CLUSTERED_PEP -w $CLUSTERED_SPLICEDEXON  -o $CLUSTERED_GFF
$TRANSCRIPT_GENOME_GFF
```

Gene expression

5 Gene expression quantification uses Salmon in mapping-based mode
https://salmon.readthedocs.io/en/latest/salmon.html#preparing-transcriptome-indices-mapping-based-mode
We quantify the expression of the cs10 RefSeq mRNA.

Software

Salmon NAME

Software

Trinity RNA-Seq NAME

Dataset

GCF_900626175.1_cs10 RNA NAME

https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_900626175.2 LINK

5.1 Create decoy-aware transcriptome file https://combine-lab.github.io/alevin-tutorial/2019/selective-alignment/. This is generated by appending the genome sequences to the transcripts sequences, and list the decoys IDs. The decoys are the genome sequences.

Command

```
TRANSCRIPT_FASTA=GCF_900626175.2_cs10_rna.fna
GENTROME_FASTA=${TRANSCRIPT_FASTA}_${REF_FASTA}
SALMON_INDEX=${GENTROME_FASTA}_index
grep "^>" $REF_FASTA | cut -d " " -f 1 | sed -i 's/>//g' >
${REF_FASTA}.decoys.txt
cat $TRANSCRIPT_FASTA $REF_FASTA  >  $GENTROME_FASTA
```

5.2 Create salmon index file

Command

```
salmon  index -p 10 -t $GENTROME_FASTA -d ${REF_FASTA}.decoys.txt -i
$SALMON_INDEX -k 31
```

5.3 Quantify each replicate

Command

```
mkdir $SALMON_DIR
lines=$(cat 21trichomes_replicates.txt)
for line in $lines
do
        a=($(echo "$line" | tr '\t' '\n'))
        mkdir $SALMON_DIR/${a[0]}.1
        mkdir $SALMON_DIR/${a[0]}.2
        mkdir $SALMON_DIR/${a[0]}.3
        salmon quant -p 5 -i ${SALMON_INDEX} -l IU -1
$FASTQTRIM_DIR/${a[1]}_1.fastq.gz -2 $FASTQTRIM_DIR/${a[1]}_2.fastq.gz
--validateMappings -o $SALMON_DIR/${a[0]}.1
        salmon quant -p 5 -i ${SALMON_INDEX} -l IU -1
$FASTQTRIM_DIR/${a[2]}_1.fastq.gz -2 $FASTQTRIM_DIR/${a[2]}_2.fastq.gz
--validateMappings -o $SALMON_DIR/${a[0]}.2
        salmon quant -p 5 -i ${SALMON_INDEX} -l IU -1
$FASTQTRIM_DIR/${a[3]}_1.fastq.gz -2 $FASTQTRIM_DIR/${a[3]}_2.fastq.gz
--validateMappings -o $SALMON_DIR/${a[0]}.3
done
```

The quantified expression is in file $SALMON_DIR/*/quant.sf

Estimate abundance

Command

```
QUANT_FILE=
ls  $SALMON_DIR/*/*quant.sf > ${QUANT_FILE}.txt

TRINITY_DIR=/path/to/trinity

${TRINITY_DIR}/util/abundance_estimates_to_matrix.pl --est_method
salmon --name_sample_by_basedir   --gene_trans_map none \
--out_prefix ${QUANT_FILE}/${QUANT_FILE} \
--quant_files ${QUANT_FILE}.txt
```

Genotype

6 SNP genotype dataset generation is described in details in separate protocols

Software

10.5281/zenodo.8351609 NAME

Biopathways

7 The biopathways module visualize the expression level or differential expression of transcripts/proteins over biological pathways. It is a web-based implemantation of MapMan https://github.com/usadellab/usadellab.github.io/tree/master/MapManJS

7.1 Download MapMan mapping files, curated pathways, and protein sequences for Arabidopsis thaliana, Eucalyptus grandis (eucalypthus), and Solanum lycopersicum (tomato)

Dataset

MapMan mappings NAME

https://mapman.gabipd.org/mapman-download LINK

Arabidopsis thaliana map
X4 Araport11
https://mapman.gabipd.org/mapmanstore?p_p_id=MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5&p_p_lifecycle=0&p_p_state=normal&p_p_mode=view&p_p_col_id=column-1&p_p_col_pos=1&p_p_col_count=2&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_Show=DownloadMapping&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_RessourceId=411&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_Download=TextMapping

Dataset

Araport11 proteins NAME

https://www.arabidopsis.org/download_files/Proteins/Araport11_protein_lists/archived/Araport11_pep_20220103.gz LINK

Eucalyptus grandis map
Egrandis_201
https://mapman.gabipd.org/mapmanstore?p_p_id=MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5&p_p_lifecycle=0&p_p_state=normal&p_p_mode=view&p_p_col_id=column-1&p_p_col_pos=1&p_p_col_count=2&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_Show=DownloadMapping&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_RessourceId=336&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_Download=TextMapping

Dataset

Egrandis_v2_0 proteins NAME

https://phytozome-next.jgi.doe.gov/info/Egrandis_v2_0 LINK

Solanum lycopersicum map

ITAG2.3

https://mapman.gabipd.org/mapmanstore?p_p_id=MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5&p_p_lifecycle=0&p_p_state=normal&p_p_mode=view&p_p_col_id=column-1&p_p_col_pos=1&p_p_col_count=2&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_Show=DownloadMapping&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_RessourceId=310&_MapManDataDownload_WAR_MapManDataDownloadportlet_INSTANCE_4Yx5_Download=TextMapping

Dataset

ITAG2.3 proteins NAME

https://solgenomics.net/ftp/tomato_genome/annotation/ITAG2.3_release/ITAG2.3_proteins_full_desc.fasta LINK

Dataset

Cannabis cs10 proteins NAME

https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/900/626/175/GCF_900626175.1_cs10/GCF_900626175.1_cs10_protein.faa.gz LINK

Software

MapManJS NAME

7.2 Align protein of cs10 vs the three plants' proteins using mmseq reciprocal best hit

Command

```
PROTEINCS10_FASTA=/path/to/cs10_pep.fasta.gz
PROTEIN1_FASTA=/path/to/ath_pep.fasta.gz
PROTEIN2_FASTA=/path/to/egr_pep.fasta.gz
PROTEIN3_FASTA=/path/to/sly_pep.fasta.gz
PAIR1_RBH=cs10_ath
PAIR2_RBH=cs10_egr
PAIR3_RBH=cs10_sly

$MMSEQ_DIR/mmseqs easy-rbh $PROTEINCS10_FASTA  $PROTEIN1_FASTA
$PAIR1_RBH tmp
$MMSEQ_DIR/mmseqs easy-rbh $PROTEINCS10_FASTA  $PROTEIN2_FASTA
$PAIR2_RBH tmp
$MMSEQ_DIR/mmseqs easy-rbh $PROTEINCS10_FASTA  $PROTEIN3_FASTA
$PAIR3_RBH tmp
```

7.3 Revise MapMan maps to use the cannabis genes using the reciprocal best hit results.

X4_ENSEMBL39_ISOFORM_Arabidop… 2MB

Egrandis_201.txt-cs10pep.out.gz 2.4MB

X4_ENSEMBL39_ISOFORM_Solanum… 1.2MB

Synteny

8 MCScanX was used to detect synteny blocks using the genes for cs10 and pkv5
https://github.com/wyp1125/MCScanX

Software

blast+ NAME

Software

MCScanX NAME

8.1 Align two reference gene models using blastp

Command

```
BLAST_DIR=/path/to/blast
PROT_FASTA_1=/path/to/cs10_pep.gz
PROT_FASTA_2=/path/to/pkv5_pep.gz  # from gene prediction results
PROT_DB1='cs10'
PROT_DB2='pkv5'
$BLAST_DIR/makeblastdb -in $PROT_FASTA_1 -dbtype prot  -out
${PROT_DB1}_db
$BLAST_DIR/blastall -i $PROT_FASTA_2 -d ${PROT_DB1}_db  -p blastp -e
1e-10 -b 5 -v 5 -m 8 -o ${PROT_DB2}-${PROT_DB1}_blastallp_b5v5m8.blast
```

8.2 From the **GFF** file, generate **TSV** file with columns *contig,gene_id,start,stop* then rename contigs into annotation code plus incremental number, ie. cs1, cs2 .. csN, pk1,pk2,..pkN. Also generate a **TSV** mapping file of the codes and original contig names. Concatenate the files for the pair.

8.3 Create a directory and move the input files. Run MCScanX

Command

```
MCSCANX_DIR=/path/to/mcscanx
PAIR_NAME=${PROT_DB1}_${PROT_DB2}
mkdir $PAIR_NAME
cp  genelist_${PAIR_NAME}.txt ${PAIR_NAME}/
cp contigmap_${PAIR_NAME}.txt ${PAIR_NAME}/
$MCSCANX_DIR/MCScanX ${PAIR_NAME}/${PAIR_NAME}
```

File ${PAIR_NAME}/${PAIR_NAME}.collinearity is generated

Tripal modules and data loaders

9 Install and enable the following Tripal modules, then load the different datasets corresponding to each module. The step-by-step guide to loading are described in each modules documentations in the provided links. The purpose of this section is to summarize the datafiles and format needed for each module, how to generate them using the results from the previous sections, or where to find them if publicly available. Datasets generated from the previous sections and needed in the next steps are compiled in https://icgrc.info/downloads

9.1 Install Tripal v3 docker

Software

Tripal v3 NAME

10 Create and load the basic information for a new genome.

10.1 Create an organism
https://tripal.readthedocs.io/en/latest/user_guide/example_genomics/organisms.html
In Chado, the organism_id is used to define a genome assembly. Multiple assemblies or versions for a species can be assigned as different "Organisms" in Tripal.
Create Genome Assembly (Analysis), and load reference genome **FASTA** file, associated to Analysis and Organism

10.2 Define the Gene Annotation (Analysis) used to generate the gene sequences.
Load and publish mRNA and protein sequences **FASTA** files generated from step 2.5 or downloaded from sources like NCBI RefSeq, associated to Analysis and Organism
https://tripal-devseed.readthedocs.io/en/latest/loading_FASTA.html

Dataset

CS10 assembly, mRNA, cds, peptide NAME

https://www.ncbi.nlm.nih.gov/assembly/GCF_900626175.2 LINK

Download cs10 sequences and GFF files

Command

```
curl -OJX GET
"https://api.ncbi.nlm.nih.gov/datasets/v2alpha/genome/accession/GCF_900
626175.2/download?
include_annotation_type=GENOME_FASTA,GENOME_GFF,RNA_FASTA,CDS_FASTA,PRO
T_FASTA,SEQUENCE_REPORT&filename=GCF_900626175.2.zip" -H "Accept:
application/zip"
```

Dataset

Purple Kush assembly v5 NAME

https://www.ncbi.nlm.nih.gov/assembly/GCA_000230575.5 LINK

Dataset

Finola assembly v2 NAME

https://www.ncbi.nlm.nih.gov/assembly/GCA_003417725.2 LINK

Steps 10.2 and 10.3 are interchangeable, provided the sequence IDs in the FASTA and GFF files should match.

10.3 Define the gene modelling analysis and load the **GFF** gene models generated in step 3.2, or downloaded from sources like NCBI RefSeq.
https://tripal-devseed.readthedocs.io/en/latest/loading_GFF.html

10.4 Load additional annotations generated from different analysis. In all analyses always generate the **XML** output format when available, since it is the most comprehensive and required by the Tripal loaders. As a rule, always read the loaders documentation first before running an analysis so that the accepted formats are generated.

Load the InterproScan result in step 3.3
https://tripal-devseed.readthedocs.io/en/latest/loading_IPS.html
and additional BLAST results performed
https://tripal-devseed.readthedocs.io/en/latest/loading_BLAST.html

11 Install Mainlab Chado Search module
https://gitlab.com/mainlabwsu/chado_search

This module uses the genome features and annotations analyses loaded in steps 2 and 3, to create the Materialize View chado_search_gene_search.

The default query definition is modified in the Materialized Views page to account for i) the nomenclature of the different annotations and references, ii) to include associated mRNA and protein annotations in gene accession and keyword search, and iii) include transitive closure when using Gene Ontology terms in keyword search. The modified SQL definition is in chado_search_gene_search.sql 6KB

12 Install Expression module
https://github.com/tripal/tripal_analysis_expression

12.1 Create the biosample **TSV** file, or download them from NCBI BioSample as **XML** is available.
For TSV, the columns should be as shown in this example
https://github.com/tripal/tripal_analysis_expression/blob/master/example_files/exampleTSV.tsv

organism should match the organism name defined in 10.1
sample_name should match previously loaded Biomaterial names if available
temp, *tissue*, *treatment* should match previously loaded Ontology terms if available

Load and publish BioSample

12.2 Load expression data generated in step 5.3, or from public expression repository like NCBI GEO.

The expression data can take a **TSV** format with sample name as column header, feature name (gene, mRNA, protein name) as row header, and expression value at each matrix element. Samples (12.1) and features (10.2) should be loaded first as described, and the names should match.
A sample expression matrix available in
https://github.com/tripal/tripal_analysis_expression/blob/master/example_files/exampleMatrix.tsv

This module creates the expression_feature_all materialized view

13 Install Phenotype module
https://analyzedphenotypes.readthedocs.io/en/latest/index.html
This module heavily relies on ontology terms to define phenotypes. There are two ways of loading ontologies supported by this module:
1. use published Bioontologies
2. use customized/specialized ontology/controlled terms
In addition, there are already loaded default ontologies in Tripal for method NCIT and unit UO

13.1 Download **OBO** files from published sources
For popular crops, use crop-specific terms from http://www.cropontology.org

In addition, use generic plant and trait terms from TO https://obofoundry.org/ontology/to and PATO https://obofoundry.org/ontology/pato.
For metabolites, use ChEBI https://purl.obolibrary.org/obo/chebi.obo

Load all OBO files necessary
https://tripal.readthedocs.io/en/latest/user_guide/example_genomics/controlled_vocabs.html

13.2 Setup Trait Ontologies using any of the loaded Ontologies
https://analyzedphenotypes.readthedocs.io/en/latest/admin_guide/setup.html
Check *Allow new terms to be added to the Controlled Vocabulary during upload* for flexibility

13.3 Upload phenotype data in **TSV** format
https://analyzedphenotypes.readthedocs.io/en/latest/user_guide/loading.html

In the tsv, the *Trait Name, Method Name, Unit* columns should match the ontology terms when available
Germplasm Accession, Germplasm Name should match preloaded Stocks when available

Stocks may also be preloaded in bulk
https://tripal.readthedocs.io/en/latest/user_guide/bulk_loader.html
or Stock https://icgrc.info/bio_data/add/37,
or Germplasm https://icgrc.info/bio_data/add/21

This module creates the mview_phenotype materialized view

14 Map viewer
https://gitlab.com/mainlabwsu/tripal_map
Maps consists of a map with and set of features (genetic markers, QTLs, physical markers)

14.1 Create map
Map https://icgrc.info/bio_data/add/16

14.2 Load map features
https://icgrc.info/admin/tripal/loaders/cmap_loader

15 Install Synteny module

https://github.com/tripal/tripal_synview

15.1 Copy the files generated from steps 8.2 and 8.3 to the tripal_synview directory

Command

```
cd tripal_synview
# organism_id 1
ORG_ID1=1
# organism_id 2
ORG_ID2=2
perl syntenyTool.pl -t mcscanx_block -c contigmap_${PAIR_NAME}.txt
${PAIR_NAME}.collinearity genelist_${PAIR_NAME} > ${PAIR_NAME}.block
perl syntenyTool.pl -t mcscanx_tripal -a $ANNOTCODE_1 -b
$ANNOTCODE_2 -c  $ORG_ID1 -d $ORG_ID2 contigmap_${PAIR_NAME}.txt
genelist_${PAIR_NAME} ${PAIR_NAME}.collinearity ${PAIR_NAME}.block  >
${PAIR}.block.4tripal.txt
```

15.2 Load the generated file ${PAIR}.block.4tripal.txt to Tripal and use in defining a new Synteny page

Multiomics queries

16 The multiomics data integration is implemented as a web-service API. The documentation in https://icgrc.info/api_doc describes the queries and applicable constraints. It is uses a Drupal module to query the various materialized views generated by the Tripal modules.

16.1 The URL queries are processed using Drupal simple_slim_api module https://www.drupal.org/project/simple_slim_api. To instantiate, install and enable the simple_slim_api module, then replace simple_slim_api.module with this customized script simple_slim_api.module 65KB , and utility script read_snps.php 13KB . Set the bcftools path and vcf files location in read_snps.php

In the module, the URL arguments are converted into SQL queries on the materialized views created by the modules in section <Tripal modules and data loaders>.

16.2 These materialize views were generated by the <Expression> and <Phenotype> modules

```
CREATE MATERIALIZED VIEW expression_feature_all AS
SELECT F.feature_id AS feature_id,
  F.name AS feature_uniquename,
  B.biomaterial_id AS biomaterial_id,
  B.name AS biomaterial_name,
  AN.analysis_id AS analysis_id,
  AN.program AS analysis_method,
  ER.signal AS signal
  FROM elementresult ER
  INNER JOIN element E ON E.element_id = ER.element_id
  INNER JOIN feature F ON F.feature_id = E.feature_id
  INNER JOIN quantification Q ON Q.quantification_id =
ER.quantification_id
  INNER JOIN acquisition AQ ON AQ.acquisition_id = Q.acquisition_id
  INNER JOIN assay A ON A.assay_id = AQ.assay_id
  INNER JOIN assay_biomaterial AB ON AB.assay_id = A.assay_id
  INNER JOIN biomaterial B ON B.biomaterial_id = AB.biomaterial_id
  INNER JOIN analysis AN ON AN.analysis_id = Q.analysis_id;

CREATE MATERIALIZED VIEW mview_phenotype AS
SELECT
  o.genus         AS organism_genus,
  trait.cvterm_id AS trait_id,
  trait.name      AS trait_name,
  proj.project_id AS project_id,
  proj.name       AS project_name,
  method.cvterm_id AS method_id,
  method.name      AS method_name,
  unit.cvterm_id   AS unit_id,
  unit.name        AS unit_name,
  loc.value       AS location,
  yr.value        AS year,
  s.stock_id      AS germplasm_id,
  s.name          AS germplasm_name,
  array_to_json(array_agg(p.value)) AS values
FROM {phenotype} p
  LEFT JOIN {cvterm} trait ON trait.cvterm_id=p.attr_id
  LEFT JOIN {project} proj USING(project_id)
  LEFT JOIN {cvterm} method ON method.cvterm_id=p.assay_id
  LEFT JOIN {cvterm} unit ON unit.cvterm_id=p.unit_id
  LEFT JOIN {stock} s USING(stock_id)
  LEFT JOIN {organism} o ON o.organism_id=s.organism_id
  LEFT JOIN {phenotypeprop} loc ON loc.phenotype_id=p.phenotype_id
AND loc.type_id = 2944
  LEFT JOIN {phenotypeprop} yr ON yr.phenotype_id=p.phenotype_id
AND yr.type_id = 141
```

```
GROUP BY
  trait.cvterm_id,
  trait.name,
  proj.project_id,
  proj.name,
  method.cvterm_id,
  method.name,
  unit.cvterm_id,
  unit.name,
  loc.value,
  yr.value,
  s.stock_id,
  s.name,
  o.genus
```

16.3 The general structure is to query triples of (DATATYPE-PROPERTY,BIOMATERIAL,VALUE). The queries as shown below don't include the WHERE clauses, which are dynamically generated from the arguments of the API call.

Query expression values, with datatype EXP

```
select feature_uniquename as property, biomaterial_id as
biomaterial, signal::text as value from
chado.expression_feature_all WHERE ...
```

Query phenotype values, with datatype PHEN

```
select  trait_name as property,  stock_id as biomaterial , values
as value  from  chado.mview_phenotype WHERE ...
```

Query biosample attributes/metadata, with datatype ID or PROP

```
select  'stock_name'as property,  stock_id biomaterial , name as
value  from chado.stock
union
select  db.name  property,  b.stock_id biomaterial , dx.accession
as value  from chado.stock b, chado.stock_dbxref bdx, chado.dbxref
dx,  chado.db
where db.db_id=dx.db_id and dx.dbxref_id = bdx.dbxref_id and
bdx.stock_id = b.stock_id
union
select 'stock_description' as property,  stock_id biomaterial ,
description as value   from chado.stock
union
select db.name  property,  b.stock_id biomaterial , dx.accession as
value from chado.stock b, chado.stock_dbxref bdx, chado.dbxref dx,
chado.db
where db.db_id=dx.db_id and dx.dbxref_id = bdx.dbxref_id and
bdx.stock_id = b.stock_id
union
select c.name  property,  b.stock_id biomaterial , bp.value  from
chado.stock b , chado.stockprop bp , chado.cvterm c
where bp.stock_id = b.stock_id and c.cvterm_id = bp.type_id
```

Query SNPs, with datatype SNP

Software	
bcftools	NAME

In the API module, call bcftools to generate the three files pos_nnnnnn.txt, samples_nnnnnn.txt and call_nnnnnn.txt using the $REGION and $SELECTED_DATASET parameters, and nnnnnn is randomly generated per request. $REGION are the genomic regions from the API argument, or from genomic regions of genes returned by keyword search in <gene_function_search>. $SELECTED_DATASET is the vcf file to use based on the reference and dataset from the API arguments.

Command

```
$BCFTOOLS/bcftools view -Oz -r ${REGIONS} -o
/tmp/mergevcf2table_nnnnnn.vcf.gz /vcfs/${SELECTED_DATASET}.vcf.gz
gatk SortVcf --CREATE_INDEX true -I  /tmp/mergevcf2table_nnnnnn.vcf.gz
-O  /tmp/mergevcf2table_nnnnnn.sorted.vcf.gz
$GATK/gatk VariantsToTable  -V
/tmp/mergevcf2table_nnnnnn.sorted.vcf.gz -O /tmp/calltable_nnnnnn.txt -
F CHROM -F POS -F REF -F ALT -GF GT
table2triple.py  /tmp/calltable_nnnnnn.txt  > /tmp/call_nnnnnn.txt
copy pos_nnnnnn from '/tmp/pos_nnnnnn.txt' with delimiter  E'\t'
$BCFTOOLS/bcftools query -f '%CHROM-%POS\n'
/tmp/mergevcf2table_nnnnnn.sorted.vcf.gz > /tmp/pos_nnnnnn.txt
$BCFTOOLS/bcftools query -l /tmp/mergevcf2table_nnnnnn.sorted.vcf.gz >
/tmp/col_nnnnnn.txt
```

Reading the generated files, create three temporary database tables pos_nnnnnn for the features, col_nnnnnn for the samples, and call_nnnnnn for the call at a feature and sample. Then the following query returns the triples of (POSITION, BIOMATERIAL,CALL).

```
select pt.pos as property,  case when s2s.samn is null then
st.sample else  s2s.samn end  as biomaterial, ct.gt as value  from
call_nnnnnn ct,  pos_nnnnnn pt,  col_nnnnnn st  left join
chado.mview_srr_prjn_samn_name s2s on s2s.srr=st.sample where
ct.colno=st.colno and ct.lineno=pt.lineno
```

chado.mview_srr_prjn_samn_name is a utility materialize view that maps the SRR,SAMN,PROJ IDs and names of the samples.

16.4 The queries from the different datasets are merged using UNION of (PROPERTY,BIOMATERIAL,VALUE). For sorting and grouping purposes, the DATATYPE column with possible values [ID, PROP, PHEN, EXP or SNP] is prepended for each PROPERTY to identify the row datatype.
First define the colpivot function (https://github.com/hnsl/colpivot) inside Postgres

Software

colpivot NAME

Pivot into table with row headers PROPERTY, column headers BIOMATERIAL, and element values VALUE

```
WITH_SQL=
UNION_SQL=
select colpivot('biomatexp_stocksample_pivoted_nnnnnn',
'$WITH_SQL $UNION_SQL ' ,
array['property'], array['accession'], '#.value', null)

/* get headers */
SELECT column_name FROM  information_schema.columns WHERE
table_name = 'biomatexp_stocksample_pivoted_nnnnnn' order by
ordinal_position

/* export table */
select distinct * from biomatexp_stocksample_pivoted_nnnnnn order
by property
```

Multiomics API Use-cases

17 These scripts demonstrate the use of the API to query various datasets and do some plotting and analysis. The web-service API documentation is available at https://icgrc.info/api_doc

The Jupyter notebook ipynb uses the client modules defined in omics_api.py, using functions defined in omics_api_utils.py and the scripts in api-modules.tgz. Static export pages of the notebook on different use cases are available at https://snp.icgrc.info/static/icgrc_omics_demo.html. Module updates are also announced/available at this page between protocol version updates.

icgrc_omics_demo.ipynb 13.1MB omics_api.py 22KB omics_api_utils.py 159KB

api-modules.tgz 33KB

Embedded non-Tripal pages

18 These tools are not part of Tripal, but are instantiated and embedded as Tripal pages in the ICGRC web portal.

18.1 Instantiate JBrowse v1 following the manual at https://jbrowse.org/docs/tutorial_classic.html

Software

JBrowse NAME

Load the genome assembly, GFF gene model tracks (from step 2, or 10.2), transcript BAM alignment and count bigwig tracks (step 4)

Command

```
bin/prepare-refseqs.pl --fasta $REF_FASTA
```

Load the GFF gene models generated in step 3.2, or downloaded from NCBI RefSeq in step 10.2

Command

```
bin/flatfile-to-json.pl --gff $GFF --trackType CanvasFeatures --
trackLabel gene_model_label
```

Add the BAM file TRANSCRIPT_BAM from step 4.2 to visualize as track.

Install and configure the Tripal-JBrowse Integration module from https://tripal-jbrowse.readthedocs.io/

18.2 The following modules use the Drupal Iframe module to embed non-Tripal modules into Tripal pages.

Download from https://www.drupal.org/node/297891 into the modules directory, extract and enable the module. Once each tools are setup in their respective directories, reference their URL address in the Iframe definition pages.

18.3 Hemp-Seek

Software	
SNP-Seek	NAME

Load VCF files generated in step 6

18.4 MapManJS

Software	
MapManJS	NAME

Place the mapping files from step 7.3 into the MapManJS MappingFiles directory, and reference them in ultramicro.html mappings[] options.

```
mappings['cs10-A.thaliana_RBH']=
"./MappingFiles/X4_ENSEMBL39_ISOFORM_ArabidopsisThaliana-
cs10pep.out";
mappings['cs10-E.grandis_RBH']="./MappingFiles/Egrandis_201.txt-
cs10pep.out";
mappings['cs10-S.lycopersicum_RBH']
="./MappingFiles/X4_ENSEMBL39_ISOFORM_SolanumLyc-cs10pep.out";
```

The customized ultramicro.html ultramicro.html 20KB