Supplementary data for "Chromosome-Level Genome Assembly and Circadian Gene Repertoire of the Patagonia blennie *Eleginops maclovinus*"

This dataset constains the genome assembly and associated annotation of the Patagonian Blennie (*Eleginops maclovinus*), extracted circadian rhythm sequences for *E. maclovinus*, other notothenenioid taxa, and teleost outgroups, as well as a copy of the bioinformatic scripts used for the assembly, annotation, and other downstream analysis. It is linked to the following publication:

> Cheng, CCH, Rivera-Colón, AG, Wilson, L, *et al.* (*in prep*) **Chromosome-Level Genome Assembly and Circadian Gene Repertoire of the Patagonia blennie *Eleginops maclovinus* - the closest ancestral proxy of Antarctic cryonotothenioids**.

Methods

An *E. maclovinus* specimen was collected from the Puerto Natales, Chile in January 2018. HMW DNA was extracted and sequenced using PacBio Sequel II and a Hi-C library. A contig-level genome assembly was first generated using `wtdgb2` (*a.k.a.* `redbean`) v2.5 (Ruan & Li 2020), and scaffolded with `juicer` v1.6.2 (Durand *et al.* 2016). PacBio and HiC raw data is available under NCBI BioProject PRJNA857989. For annotation, the RNA-seq data generated by Bilyk *et al.* (2018) was aligned to the genome, and processed using `BRAKER` v2.1.6 Brůna *et al.* 2021. The generated annotation was then further processed using `TSEBRA` v1.0.1 (Gabriel *et al.* 2021). Using a custom Python script (see scripts section), we curated the `TSEBRA` output to guarantee consistency in the naming of genes and transcripts, as well as incorporating gene names and description based on the corresponding zebrafish orthologs.

A conserved synteny analysis using `synolog` (Catchen *et al.* 2009; Small *et al.* 2016) was employed for the manual curation of the assemblies. For example, we identifying missassemblies in structural variants limited to contig boundaries or merged scaffolds belonging to the same chromosome sequences. We used a custom Python program to propagate these changes through the constituent assembly files.

For the circadian rhythm comparative analysis, assemblies and annotation were downloaded from genomic databases (e.g., ENSEMBL, NCBI). Circadian gene orthologs were identified using `synolog` and extracted using custom Python scripts.

A detailed step-by-step description of the methods is available on the publication (Cheng et al. *in prep*).

Usage Notes

All assembly and annotation files are gzipped, but are otherwise standard bioinformatic formats (i.e., FASTA for genome assembly and coding/amino acid sequences, GTF for annotation, AGP for scaffolding). In addition, bioinformatic scripts for data generation and analysi are in Python (`*.py`) or Bash (`*.sh`, but might require the installation of additional, open-source software (e.g., `wtdbg2`, `BRAKER`)

See links for a description of the FASTA (http://www.ncbi.nlm.nih.gov/blast/fasta.shtml), and GTF (https://useast.ensembl.org/info/website/upload/gff.html), and AGP (https://www.ncbi.nlm.nih.gov/assembly/agp/AGP_Specification/) file format specifications.

Genome assembly and annotation

Files for the *de novo* assembly and annotation of *E. maclovinus*. Files have the label `emac.rtc.rv5`, which denotes the species (`emac`), the annotation (`rtc`, `redbean`+`TSEBRA` curated), and integration (`rv5`, `redbean` assembly, 5th iteration).

Due to their size, files are compressed in a tarball. To extract, do:

```
tar -xzvf emac_asm_annot.tar.gz
```

The resulting directory contains the following data:

File description

File *	Description
`emac.rtc.rv5.fa`	Genome assembly in nucleotide FASTA format.
`emac.rtc.rv5.agp`	Assembly structure in AGP format.
`emac.rtc.rv5.gtf`	Genome annotation in GTF format.
`emac.rtc.rv5.cds.fa`	Genomic sequence for all annotated protein-coding genes in nucleotide FASTA format.
`emac.rtc.rv5.protein.fa`	Protein sequence for all annotated protein-coding genes in amino acid FASTA format.

* Does not include the gzipped compression suffix (`.gz`)

Circadian Rhythm Orthologs Analysis

Sequence and annotation files for circadian rhythm gene orthologs across notothenioids and teleost orthologs.

Due to their size and to preserve organization in directories, files are compressed in a tarball. To extract, do:

```
tar -xzvf circadian_orthologs.tar.gz
```

Assemblies used

Scientific Name	Common Name	Assembly Label	NCBI/Ensembl ID	NCBI Accession
Eleginops maclovinus	Patagonia Blennie	emac.rtc.rv5	NA	NA
Chaenocephalus aceratus	Blackfin icefish	cace.kuc.kuc	KU_Ca_2.0	GCA_023974075.1

Scientific Name	Common Name	Assembly Label	NCBI/Ensembl ID	NCBI Accession
Champsocephalus esox	Pike icefish	ceso.ftc.fv8	NA	NA
Cottoperca gobio	Channel Bull Blennie	cgob.def.def	fCotGob3.1	GCF_900634415.1
Champsocephalys gunnari	Mackerel icefish	cgun.ftc.fv8	NA	NA
Dissostichus mawsoni	Antarctic toothfish	dmaw.def.def	KU_Dm_1.0	GCA_011823955.1
Gymnodraco acuticeps	Antarctic ploughfish	gyac.def.def	fGymAcu1.1	GCF_902827175.1
Harpagifer antarcticus	Antarctic spiny plunderfish	hant.def.def	fHarAnt1.1	GCA_902827135.1
Pogonophryne albipinna	Whitefin plunderfish	palb.def.def	KU_S6	GCA_028583405.1
Pseudochaenichthys georgianus	South Georgia icefish	pgeo.def.def	fPseGeo1.1	GCF_902827115.1
Trematomus bernacchii	Emerald rockcod	tber.def.def	fTreBer1.1	GCF_902827165.1
Trematomus loennbergii	Scaly rockcod	tloe.def.def	KUTl01	NA
Danio rerio	Zebrafish	drer.def.def	GRCz11	GCF_000002035.6

File description

For each species, the data for the circadian ortholog sequences and annotations is stored in its own separate directory, named according to its corresponding assembly label shown in the table above. Each directory contains the following files:

File *	Description
`<label>.cds.fa`	Sequence in nucleotide FASTA format.
`<label>.gtf`	Coordinates and genomic annotation in GTF format.
`<label>.peptide.fa`	Sequence in amino acid FASTA format.

* `<label>` is the assembly label and directory name for the species.

For example, the following files are available for *E. maclovinus*:

```
$ ls -1 emac.rtc.rv5/
emac.rtc.rv5.cds.fa
emac.rtc.rv5.gtf
emac.rtc.rv5.peptide.fa
```

E. maclovinus Iso-Seq data

The *E. maclovinus* fin transcriptome was sequenced using PacBio Iso-Seq in order to confirm expression of circadian rhythm orthologs. The consensus sequences for this transcriptome are available in nucleotide FASTA format in the file `emac_isoseq_transcripts/emac_isoseq.fa`.

Bioinformatic scripts

Genome assembly

Subsample reads

`sample_reads.py`:

Custom Python script to downsample raw FASTQ files to a given coverage and size distribution. Script further described in Rayamajhi *et al.* 2022.

Usage:

```
$ ./sample_reads.py -h
usage: sample_reads.py [-h] [-1 SE_PATH] [-2 PE_PATH] [-o OUT] [-r
FRACTION]
                       [-g GENOME] [-c DEPTH] [-l LENGTH]
                       [--max_length MAX_LENGTH] [--seed SEED]

Subsample a set of reads from one or a pair of FASTQ files. One can either
select a fraction of total reads to sample, or one can specify a genome
length
and depth of coverage to sample to fill.

optional arguments:
  -h, --help            show this help message and exit
  -1 SE_PATH, --se_path SE_PATH
                        Path to single-end reads.
  -2 PE_PATH, --pe_path PE_PATH
                        Path to paired-end reads.
  -o OUT, --out OUT     Path to write sampled data.
  -r FRACTION, --fraction FRACTION
                        Fraction of reads you want to randomly sample.
  -g GENOME, --genome GENOME
                        Approximate size of the genome for use in
determining
                        depth of coverage (in basepairs, or with G, M, or
K
                        suffix).
  -c DEPTH, --depth DEPTH
                        Depth of coverage to randomly sample to (based on
                        specified genome size).
  -l LENGTH, --length LENGTH
                        Only consider reads longer than length limit for
                        sampling (in basepairs, or with G, M, or K
suffix).
  --max_length MAX_LENGTH
                        Only consider reads shorter than max_length limit
for
                        sampling (in basepairs, or with G, M, or K
suffix).
  --seed SEED           Specify a seed to the random number generator to
start
                        this sampling.
```

Readbean genome assembly

`wtdbg2.sh`:

Generate a contig-level genome assembly from the downsampled PacBio CLR reads using the *RedBean* assembler. First, call the assembler `wtdbg2`, then generate a FASTA consensus with `wtpoa-cns`.

Self-correct assembly

`arrow.sh`:

Run a self-polish of a genome assembly with the raw (unprocessed) PacBio CLR reads using the *Arrow* pipeline.

Hi-C scaffolding

`juicer.sh`:

Scaffold the contig-level genome assembly using *Juicer*. It first aligns the Illumina Hi-C with *BWA* `mem`, processing alignments with *SAMtools*. Then, it runs the *Juicer* pipeline to identify the location of *DpnII* cutsites, identifying Hi-C junctions with `juicer.sh`, and scaffolding with `3d-dna`.

Contiguity statistics

`quast.sh`:

Assess the continuity statistics for a genome assembly using *Quast*.

Gene completeness

`busco_v5.sh`:

Assess the gene-completeness of a genome assembly using *BUSCO* v5. Use the `actinopterygii_odb10` lineage and zebrafish protein sequences for comparison in *Augustus*.

Manually alter genome

`alter_genome_structure.py`:

Custom Python script to alter the structure of the genome assembly. The script is uses to implement manual corrections to the assembly, e.g., inverting specific contigs, performing splits or merges, renaming scaffolds, etc.

Usage:

```
$ ./alter_genome_structure.py -h
usage: alter_genome_structure.py [-h] --chromosomes CHR_AGP_PATH
                                 [--scaffolds SCAF_AGP_PATH] [-f
FASTA_PATH]
                                 [--gff GFF_PATH] [--gtf GTF_PATH] -a
                                 ALTER_PATH -o OUT_PATH [--prefix
FILE_PREFIX]
                                 [--synthetic SYNTHETIC_CHR]
                                 [--linelen LINELEN] [--gapsize GAPSIZE]
                                 [-w WL] [--contigs CTG_AGP_PATH]

Alter the structure of a genome assembly, including the AGP, GFF/GTF, and
FASTA files by following defined changes listed in the ALTER file.

optional arguments:
  -h, --help            show this help message and exit
  --chromosomes CHR_AGP_PATH
                        AGP file describing the scaffolds within each
                        chromosome.
  --scaffolds SCAF_AGP_PATH
                        AGP file describing unplaced scaffolds. If
specified,
                        these will be added to the main chromosome AGP
objects
                        for processing.
  -f FASTA_PATH, --fasta FASTA_PATH
                        Path to assembly FASTA genome sequence file that
will
                        be modifide to implement structural changes.
  --gff GFF_PATH        Path to GFF file describing gene annotations that
will
                        be modified to implement structural changes.
  --gtf GTF_PATH        Path to GTF file describing gene annotations that
will
                        be modified to implement structural changes.
  -a ALTER_PATH, --alter ALTER_PATH
                        Path to file containing alterations to make to the
                        genome.
  -o OUT_PATH, --out OUT_PATH
                        Write altered files to this path.
  --prefix FILE_PREFIX  Name output files to OUT_PATH naming them with
this
                        common prefix [default: original file name date
                        stamped].
  --synthetic SYNTHETIC_CHR
                        Sometimes unordered scaffolds are concatenated
into a
                        synthetic chromosome. If so, give the name of that
                        chromosome here (must match FASTA/AGP IDs).
  --linelen LINELEN     Line length for printing FASTA sequences [default:
                        60bp].
  --gapsize GAPSIZE     When inserting a new gap between scaffolds make it
                        this size [default: 500bp].
  -w WL, --whitelist WL
                        Process a single chromosome specified here.
  --contigs CTG_AGP_PATH
                        AGP file describing the contigs within each
chromosome
                        [rare; use in addition to the --chromosomes
option].
```

Genome annotation

Align RNA-seq reads

`star_alignment.sh`:

Generate an index of the reference assembly using *STAR* `genomeGenerate` and then align the paired-end RNAseq reads using *STAR* `alignReads`.

Model Repeats

`repeat_modeler.sh`:

Use *RepeatModeler* to build a repeat database for a target reference assembly.

Mask Repeats

`repeat_masker.sh`:

Use *RepeatMasker* to mask repetitive sequences in a target reference assembly. Script used the repeats generated by *RepeatModeler* as inputs, in combination with a *RepBase* Teleost-specific repeat library.

Protein-based annotation

`braker_protein.sh`:

Run *BRAKER* in protein mode, using OrthoDB10 reference protein as an input, to annotate protein-coding genes in the target reference assembly.

Transcript-based annotation

`braker_transcript.sh`:

Run *BRAKER* in transcript mode, using aligned RNAseq reads as input, to annotate protein-coding genes in the target reference assembly.

Merge annotations

`tsebra.sh`:

Use *TSEBRA* to merge the *BRAKER* protein and *BRAKER* transcript annotations. It exports gene annotations (in GTF) that are supported by both methods. Then, with the new GTF it runs *Agustus* `getAnnoFastaFromJoingenes.py` to get CDS and protein FASTAs.

Correct TSEBRA output

`correct_tsebra_annots.py`:

Custom Python script used to make the naming of the resulting *TSEBRA* annotations uniform across GTF, CDS FASTA and Protein FASTA files. If available, if can also add the corresponding functional annotation from *InterProScan* and the Zebrafish gene names to the GTF annotations. It can also remove "rogue" annotations (e.g., `exon` fields without parent `gene`).

Usage:

```
$ ./correct_tsebra_annots.py -h
usage: correct_tsebra_annots.py [-h] -g GTF [-c CDS] [-a FAA] [-o OUTD]
                                [-i IPR] [-m HOMOLOGS] [-b BASENAME]
                                [--keep-orig-transcript-ids]

optional arguments:
  -h, --help            show this help message and exit
  -g GTF,  --gtf GTF     TSEBRA Annotation GTF
  -c CDS,  --cds CDS     TSEBRA CDS FASTA
  -a FAA,  --faa FAA     TSEBRA Amino Acid FASTA
  -o OUTD, --outd OUTD   Output Directory
  -i IPR,  --ipr IPR     InterProScan Output Table
  -m HOMOLOGS, --homologs HOMOLOGS
                        Synolog Zebrafish Homologs
  -b BASENAME, --basename BASENAME
                        Basename for Files
  --keep-orig-transcript-ids
                        Keep the original TSEBRA transcript IDs in all
files.
```

Synteny analysis

Make BLAST database

`make_blast_db.sh`:

Make a *BLAST* database for a given annotation, using the protein FASTA file.

Run BLASTs

`run_reciprocal_blast.sh`:

Run a protein *BLAST* (`blastp`) for a given query/subject species pair. Uses the *BLAST* database generated by `run_reciprocal_blast.sh` as an subject input, and the other species' protein FASTA as a query.

Run synteny analysis

`run_synolog.sh`:

Run a conserved synteny analysis using *Synolog*. For a list of given species, *Synolog* requires reciprocal *BLAST* outputs for each species' pair, as well as annotations (in GTF/GFF) for each genome.

Processing circadian rhythm orthologs

Find target orthologs IDs

`extract_synolog_homologs.py`:

Custom Python script that finds the orthologs genes in a query species based on a set of reference genes. The sequences are extracted based on the orthology determined by *Synolog*.

Usage:

```
$ extract_synolog_homologs.py -h
usage: extract_synolog_homologs.py [-h] -r REFERENCE_HOMOLOGS -s
                                   SYNOLOG_HOMOLOGS_TSV -o OUT_DIR [-i
REF_ID]

optional arguments:
  -h, --help            show this help message and exit
  -r REFERENCE_HOMOLOGS, --reference-homologs REFERENCE_HOMOLOGS
                        File with the reference IDs
  -s SYNOLOG_HOMOLOGS_TSV, --synolog-homologs-tsv SYNOLOG_HOMOLOGS_TSV
                        Path to the XXX-YYY_homologs.tsv
  -o OUT_DIR, --out-dir OUT_DIR
                        Output directory
  -i REF_ID, --ref-id REF_ID
                        ID of ref annotation in the homologs file
```

Extract sequence and annotation for the orthologs

Custom Python script to extract the sequence and annotation of a set of target genes, which could be identified using `extract_synolog_homologs.py`, from the genome-wide annotation GTF and the protein/amino acid FASTA.

Usage:

```
$ ./extract_target_genes.py -h
./extract_target_genes.py -h

usage: extract_target_genes.py [-h] -t TARGET_GENES [-o OUT_DIR] -g GTF -p
                               PROTEIN_FA [-c CDS_FA] -n ORG_NAME -a
                               ASSEMBLY_ID

optional arguments:
  -h, --help            show this help message and exit
  -t TARGET_GENES, --target-genes TARGET_GENES
                        Target Gene Table
  -o OUT_DIR, --out-dir OUT_DIR
                        Output Directory
  -g GTF, --gtf GTF     GTF Annotation
  -p PROTEIN_FA, --protein-fa PROTEIN_FA
                        Protein FASTA
  -c CDS_FA, --cds-fa CDS_FA
                        CDS FASTA
  -n ORG_NAME, --org-name ORG_NAME
                        Name of the Organism
  -a ASSEMBLY_ID, --assembly-id ASSEMBLY_ID
                        Assembly ID (used to name outputs)
```

Additional data availability

Raw PacBio CLR and Hi-C Illumina reads for the *E. maclovinus* genome assembly are available under NCBI BioProject PRJNA917608. Raw RNAseq reads used for annotation are indexed under NCBI Bioproject PRJNA368682 and were originally published in Bilyk *et al.* (2018).

Notes

2023-04-14

Initial compilation of files and submission to DRYAD.

Authors

Angel G. Rivera-Colón
Department of Evolution, Ecology, and Behavior
University of Illinois at Urbana-Champaign
angelgr2@illinois.edu

Julian M. Catchen
Department of Evolution, Ecology, and Behavior
University of Illinois at Urbana-Champaign
jcatchen@illinois.edu

C-H Christina Cheng
Department of Evolution, Ecology, and Behavior

University of Illinois at Urbana-Champaign
c-cheng@illinois.edu