phyluce documentation

Release 1.7.0

Brant C. Faircloth

Mar 04, 2021

Contents:

Release v1.7.0

Author Brant C. Faircloth

Date 04 March 2021 20:11 UTC (+0000)

phyluce (phy-**loo**-chee) is a software package that was initially developed for analyzing data collected from ultraconserved elements in organismal genomes (see *References* and http://ultraconserved.org for additional information).

The package includes a number of tools spanning:

- the assembly of raw read data to contigs
- the separation of UCE loci from assembled contigs
- parallel alignment generation, alignment trimming, and alignment data summary methods in preparation for analysis
- SNP calling and contig correction using raw-read data

As it stands, the phyluce package is useful for analyzing both data collected from UCE loci and also data collection from other types of loci for phylogenomic studies at the species, population, and individual levels.

CHAPTER 1

Contributions

phyluce is open-source (see *License*) and we welcome contributions from anyone who is interested. Please make a pull request on github.

CHAPTER 2

Issues

The issue tracker for phyluce is available on github. If you have an issue, please ensure that you are experiencing this issue on a supported OS (see *Installation*) using the conda installation of phyluce. When submitting issues, please include a test case demonstrating the issue and indicate which operating system and phyluce version you are using.

CHAPTER 3

Guide

3.1 Purpose

Phylogenomics offers the possibility of helping to resolve the Tree of Life. To do this, we often need either very cheap sources of organismal genome data or decent methods of subsetting larger genomes (e.g., vertebrates; 1 Gbp) such that we can collect data from across the genome in an efficient and economical fashion, find the regions of each genome that are shared among organisms, and attempt to infer the evolutionary history of the organisms in which we're interested using the data we collect.

Genome reduction techniques offer one way to collect these types of data from both small- and large-genome organisms. These "reduction" techniques include various flavors of amplicon sequencing, RAD-seq (**R**estriction site **A**ssociated **D**NA markers), RNA-seq (transcriptome sequencing), and sequence capture methods.

phyluce is a software package for working with data generated from sequence capture of UCE (**u**ltra-**c**onserved **e**lement) loci, as first published in [BCF2012]. Specifically, phyluce is a suite of programs to:

- assemble raw sequence reads from Illumina platforms into contigs
- determine which contigs represent UCE loci
- filter potentially paralagous UCE loci
- generate different sets of UCE loci across taxa of interest

Additionally, phyluce is capable of the following tasks, which are generally suited to any number of phylogenomic analyses:

- produce large-scale alignments of these loci in parallel
- manipulate alignment data prior to further analysis
- convert alignment data between formats
- compute statistics on alignments and other data

phyluce is written to process data/individuals/samples/species in parallel, where possible, to speed execution. Parallelism is achieved through the use of the Python multiprocessing module, and most computations are suited to workstation-class machines or servers (i.e., rather than clusters). Where cluster-based analyses are needed, phyluce will produce the necessary outputs for input to the cluster/program that you are running so that you can setup the run according to your cluster design, job scheduling system, etc. Clusters are simply too heterogenous to do a good job at this part of the analytical workflow.

3.1.1 Longer-term goals (v2.0.0+ and beyond)

We are also working towards adding:

- simplify the CLI (command-line interface) of phyluce
- add additioanl `workflows` for multi-step analyses

3.1.2 Who wrote this?

This documentation was written primarily by Brant Faircloth (http://faircloth-lab.org). Brant is also responsible for the development of most of the phyluce code. Bugs within the code are usually his.

You can find additional authors and contributors in the *Attributions* section.

3.1.3 How do I report bugs?

To report a bug, please post an issue to https://github.com/faircloth-lab/phyluce/issues. Please also ensure that you are using one of the "supported" operating systems and a supported installation method. Please see the *Installation* section for more details.

3.2 Installation

phyluce uses a number of tools that allow it to assemble data, search for UCE loci, align results reads, manipulate alignments, prepare alignments for analysis, etc. To accomplish these goals, phyluce uses wrappers around a number of programs that do each of these tasks (sometimes phyluce can use several different programs that accomplish the same task in different ways). As a result, the dependency chain (the programs that phyluce requires to run) is reasonably complex.

In the current versions (> 1.7.x), **and we very strongly suggest** that users install phyluce using the miniconda Python distribution.

Attention: We do not support installing phyluce through means other than the conda installer. This means that we do not test phyluce against any binaries, other than those we build and distribute through conda. Although you can configure phyluce to use binaries of different provenance, this is not officially supported.

Note: We build and test the binaries available through conda using 64-bit operating systems that include:

- MacOS 10.15
- Ubuntu 20.04 LTS

phyluce is also available for use as a docker image. Underneath the hood the docker image runs Ubuntu 20.04 LTS and installs phyluce and related packages using conda.

3.2.1 Install Process

Attention: We do not support phyluce on Windows, although you technically should be able to install phyluce on Windows using the Windows Subsystem for Linux (WSL) and installing Ubuntu 20.04 LTS from the Windows Store. You should also be able to use the docker image.

Using Conda

Note: We build and test the binaries available through conda using 64-bit operating systems that include the following. We will officially support MacOS 10.16 when the github build system offers this platform for automated tests.

- MacOS 10.15
- Ubuntu 20.04 LTS

The installation process is a 2-step process. You need to:

1. Install miniconda
2. Install phyluce

Installing phyluce will install all of the required binaries, libraries, and Python dependencies that you need to run the program.

Install miniconda

First, you need to install miniconda. Follow the instructions for your platform that are available from conda.io. After you have run the install process **be sure** that you:

1. close and re-open your terminal window
2. run `conda list` which should produce output

Install phyluce

Current practice with conda is to keep all environments separate and **not** to use the base environment as a "default" environment. So, we will be installing phyluce into an environment named `phyluce-vX.X` where the `X.x` represents the version you choose.

1. Go to the phyluce github release page
2. Download the appropriate `*.yml` file for the phyluce version you want and the **operating system you are using**
3. Install that into an environment corresponding to the phyluce version, e.g. `phyluce-1.7` following the instructions on the phyluce release page

This will create an environment named `phyluce-X.x`, then download and install everything you need to run phyluce into this `phyluce-X.x` conda environment.

To use your new phyluce environment, you **must** run (replace `X.x` with the correct version):

```
conda activate phyluce-X.x
```

To stop using this phyluce environment, you **must** run:

```
conda deactivate
```

What conda installs

When you install phyluce, it specifies a number of dependencies that it needs to run. If you would like to know **everything** that conda has installed, you can open up the `*.yml` you downloaded (it is simply a text file) and take a look at the contents.

From within the conda environment, you can also run

```
conda activate phyluce-X.x
conda list
```

Added benefits

An added benefit of using conda is that you can also run all of the 3rd-party binaries without worrying about setting the correct $PATH, etc.

For example, phyluce requires MUSCLE for installation, and MUSCLE was installed by conda as a dependency of phyluce. Because conda puts all of these binaries in our `$PATH` when the environment is activateed, we can also just run MUSCLE on the command-line, with, e.g.,:

```
$ muscle -version

MUSCLE v3.8.1551 by Robert C. Edgar
```

Using Docker

We also provide phyluce as a docker image, which means you can run the phyluce installation anywhere that you can run docker. The docker image is built on Ubuntu 20.04 LTS using conda. To pull the docker image:

1. Go to the phyluce github release page
2. Find the phyluce release you want (usually the most recent)
3. Run the `docker pull` command listed

Although using docker is beyond the scope of this guide, you can run phyluce within a docker using a command similar to the following, e.g.:

```
docker run fairclothlab/phyluce:<tag> phyluce <phyluce_program_name>
```

Where `<tag>` corresponds to the version of phyluce you are using. When you run this, all commands are run in the default directory `/work` and the user within the container is named `phyluce`.

You will very likely want to mount a local directory (on your computer) to this `/work` directory in the docker container and make yourself the owner of the result files. If you are working in `/home/you/phyluce` on your computer, you can accomplish all of by running phyluce like:

```
docker run \
    -v /home/you/phyluce:/data \
    --user $(id -u):$(id -g) \
    fairclothlab/phyluce:1.7.0 \
    phyluce_assembly_assemblo_spades \
```

(continues on next page)

(continued from previous page)

```
    --output spades-test \
    --config assembly.conf \
    --cores 12
```

The `-v /home/you/phyluce:/data` maps your directory (`/home/you/phyluce`) onto the container working directory (`/work`), the `--user $(id -u):$(id -g)` makes the owner of the files in the container your user and group, the `fairclothlab/phyluce:1.7.0` is the name of the image to use, and the rest are standard phyluce command.

Finally, you may want to run many commands in the docker container (e.g. as in an entire analysis run). This can be accomplished by starting a **bash_** shell in the container, and working from within the container's bash prompt, as in:

```
docker run \
    -v /home/you/phyluce:/data \
    --user $(id -u):$(id -g) \
    -i -t fairclothlab/phyluce:1.7.0 \
    /bin/bash

# this drops you into the shell, where you can run commands, e.g.:
@d51aa2f2d565:/data$
```

Using Singularity

If you are using Singularity, you should be able to pull the Docker image, and convert it for use, although this is not tested and is not supported. For example:

```
singularity pull docker://fairclothlab/phyluce:1.7.0
```

If that does not work, you could also use the phyluce Dockerfile to create a Singularity definition file, and build a Singularity image.

3.2.2 phyluce configuration

As of v1.5.x, phyluce uses a configuration file to keep track of paths to relevant binaries, as well as some configuration information. This file is located at `$CONDA_PREFIX/phyluce/config`. Although you can edit this file directly, you can also create a user-specific configuration file at *~/.phyluce.conf* (**note the preceding dot**), which will override the default values with different paths.

So, if you need to use a slightly different binary or you want to experiment with new binaries (e.g. for assembly), then you can change the paths in this file rather than deal with hard-coded paths.

Attention: This **WILL NOT** work for the docker image by default. You also do NOT **need** to to anything with this file - $PATHs should automatically resolve.

Warning: Changing the *$PATHs* in the config file can break things pretty substantially, so please use with caution. If you are making changes, edit the copy at `~/.phyluce.conf`) rather than the default copy.

The format of the config file as of v1.7 looks similar to the following:

```
[binaries]
abyss:$CONDA/bin/ABYSS
abyss-pe:$CONDA/bin/abyss-pe
bcftools:$CONDA/bin/bcftools
bedtools:$CONDA/bin/bedtools
bwa:$CONDA/bin/bwa
gblocks:$CONDA/bin/Gblocks
lastz:$CONDA/bin/lastz
mafft:$CONDA/bin/mafft
muscle:$CONDA/bin/muscle
pilon:$CONDA/bin/pilon
raxml-ng:$CONDA/bin/raxml-ng
samtools:$CONDA/bin/samtools
seqtk:$CONDA/bin/seqtk
spades:$CONDA/bin/spades.py
trimal:$CONDA/bin/trimal
velvetg:$CONDA/bin/velvetg
velveth:$CONDA/bin/velveth
snakemake:$CONDA/bin/Snakemake

[workflows]
mapping:$WORKFLOWS/mapping/Snakefile
correction:$WORKFLOWS/contig-correction/Snakefile
phasing:$WORKFLOWS/phasing/Snakefile

#----------------
#    Advanced
#----------------

[headers]
trinity:comp\d+_c\d+_seq\d+|c\d+_g\d+_i\d+|TR\d+\|c\d+_g\d+_i\d+|TRINITY_DN\d+_c\d+_
↪g\d+_i\d+
velvet:node_\d+
abyss:node_\d+
idba:contig-\d+_\d+
spades:NODE_\d+_length_\d+_cov_\d+.\d+

[spades]
max_memory:4
cov_cutoff:5
```

3.2.3 Other useful tools

You will need to be familiar with the command-line/terminal, and it helps to have a decent text editor for your platform. Here are some suggestions that are free:

- vscode
- atom

3.3 Phyluce Tutorials

3.3.1 Tutorial I: UCE Phylogenomics

In the following example, we are going to process raw read data from UCE enrichments performed against several divergent taxa so that you can get a feel for how a typical analysis goes. For more general analysis notes, see the *UCE Processing for Phylogenomics* chapter. That said, this is a good place to start.

The taxa we are working with will be:

- Mus musculus (PE100)
- Anolis carolinensis (PE100)
- Alligator mississippiensis (PE150)
- Gallus gallus (PE250)

Download the data

You can download the data from figshare (http://dx.doi.org/10.6084/m9.figshare.1284521). If you want to use the command line, you can use something like:

```
# create a project directory
mkdir uce-tutorial

# change to that directory
cd uce-tutorial

# download the data into a file names fastq.zip
wget -O fastq.zip https://ndownloader.figshare.com/articles/1284521/versions/2

# make a directory to hold the data
mkdir raw-fastq

# move the zip file into that directory
mv fastq.zip raw-fastq

# move into the directory we just created
cd raw-fastq

# unzip the fastq data
unzip fastq.zip

# delete the zip file
rm fastq.zip

# you should see 6 files in this directory now
ls -l

-rw-r--r--. 1 bcf users 4.4M Feb 22 14:14 Alligator_mississippiensis_GGAGCTATGG_L001_
↪R1_001.fastq.gz
-rw-r--r--. 1 bcf users 4.3M Feb 22 14:14 Alligator_mississippiensis_GGAGCTATGG_L001_
↪R2_001.fastq.gz
-rw-r--r--. 1 bcf users 4.9M Feb 22 14:14 Anolis_carolinensis_GGCGAAGGTT_L001_R1_001.
↪fastq.gz
-rw-r--r--. 1 bcf users 4.9M Feb 22 14:15 Anolis_carolinensis_GGCGAAGGTT_L001_R2_001.
↪fastq.gz
```

(continues on next page)

(continued from previous page)

```
-rw-r--r--. 1 bcf users 7.6M Feb 22 14:15 Gallus_gallus_TTCTCCTTCA_L001_R1_001.fastq.
↪gz
-rw-r--r--. 1 bcf users 8.4M Feb 22 14:15 Gallus_gallus_TTCTCCTTCA_L001_R2_001.fastq.
↪gz
-rw-r--r--. 1 bcf users 4.9M Feb 22 14:16 Mus_musculus_CTACAACGGC_L001_R1_001.fastq.gz
-rw-r--r--. 1 bcf users 4.9M Feb 22 14:16 Mus_musculus_CTACAACGGC_L001_R2_001.fastq.gz
```

Alternatively, if you think of the filesystem as a tree-like structure, the directory in which we are working (*uce-tutorial*) would look like:

```
uce-tutorial
+── raw-fastq
    +── Alligator_mississippiensis_GGAGCTATGG_L001_R1_001.fastq.gz
    +── Alligator_mississippiensis_GGAGCTATGG_L001_R2_001.fastq.gz
    +── Anolis_carolinensis_GGCGAAGGTT_L001_R1_001.fastq.gz
    +── Anolis_carolinensis_GGCGAAGGTT_L001_R2_001.fastq.gz
    +── Gallus_gallus_TTCTCCTTCA_L001_R1_001.fastq.gz
    +── Gallus_gallus_TTCTCCTTCA_L001_R2_001.fastq.gz
    +── Mus_musculus_CTACAACGGC_L001_R1_001.fastq.gz
    +── Mus_musculus_CTACAACGGC_L001_R2_001.fastq.gz
```

If you do not want to use the command line, you can download the data using the figshare interface or by clicking:

http://downloads.figshare.com/article/public/1284521

Count the read data

Usually, we want a count of the actual number of reads in a given sequence file for a given species. We can do this several ways, but here, we'll use tools from unix, because they are fast. The next line of code will count the lines in each R1 file (which should be equal to the reads in the R2 file) and divide that number by 4 to get the number of sequence reads.

```
for i in *_R1_*.fastq.gz; do echo $i; gunzip -c $i | wc -l | awk '{print $1/4}'; done
```

You should see:

```
Alligator_mississippiensis_GGAGCTATGG_L001_R1_001.fastq.gz
50000
Anolis_carolinensis_GGCGAAGGTT_L001_R1_001.fastq.gz
50000
Gallus_gallus_TTCTCCTTCA_L001_R1_001.fastq.gz
50000
Mus_musculus_CTACAACGGC_L001_R1_001.fastq.gz
50000
```

Notice that all the read counts are equal - that is because these 50,000 reads in each R1 and R2 file were subsampled, randomly, from a file of many more reads.

Clean the read data

The data you just downloaded are actual, raw, untrimmed fastq data. This means they contain adapter contamination and low quality bases. We need to remove these - which you can do several ways. We'll use another program that I wrote (illumiprocessor) because it allows us to trim many different indexed adapters from individual-specific fastq files - something that is a pain to do by hand. That said, you can certainly trim your reads however you would like. See the illumiprocessor website for instructions on installing the program.

To use this program, we will create a configuration file that we will use to inform the program about which adapters are in which READ1 and READ2 files. The data we are trimming, here, are from TruSeq v3 libraries, but the indexes are 10 nucleotides long. We will set up the trimming file with these parameters, but please see the illumiprocessor documentation for other options.

```
# this is the section where you list the adapters you used.  the asterisk
# will be replaced with the appropriate index for the sample.
[adapters]
i7:AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT

# this is the list of indexes we used
[tag sequences]
BFIDT-166:GGAGCTATGG
BFIDT-016:GGCGAAGGTT
BFIDT-045:TTCTCCTTCA
BFIDT-011:CTACAACGGC

# this is how each index maps to each set of reads
[tag map]
Alligator_mississippiensis_GGAGCTATGG:BFIDT-166
Anolis_carolinensis_GGCGAAGGTT:BFIDT-016
Gallus_gallus_TTCTCCTTCA:BFIDT-045
Mus_musculus_CTACAACGGC:BFIDT-011

# we want to rename our read files something a bit more nice - so we will
# rename Alligator_mississippiensis_GGAGCTATGG to alligator_mississippiensis
[names]
Alligator_mississippiensis_GGAGCTATGG:alligator_mississippiensis
Anolis_carolinensis_GGCGAAGGTT:anolis_carolinensis
Gallus_gallus_TTCTCCTTCA:gallus_gallus
Mus_musculus_CTACAACGGC:mus_musculus
```

I create this file in a directory **above** the one holding my reads, so the structure looks like:

```
uce-tutorial
+-- illumiprocessor.conf
+-- raw-fastq
    +-- Alligator_mississippiensis_GGAGCTATGG_L001_R1_001.fastq.gz
    +-- Alligator_mississippiensis_GGAGCTATGG_L001_R2_001.fastq.gz
    +-- Anolis_carolinensis_GGCGAAGGTT_L001_R1_001.fastq.gz
    +-- Anolis_carolinensis_GGCGAAGGTT_L001_R2_001.fastq.gz
    +-- Gallus_gallus_TTCTCCTTCA_L001_R1_001.fastq.gz
    +-- Gallus_gallus_TTCTCCTTCA_L001_R2_001.fastq.gz
    +-- Mus_musculus_CTACAACGGC_L001_R1_001.fastq.gz
    +-- Mus_musculus_CTACAACGGC_L001_R2_001.fastq.gz
```

Now I run illumiprocessor against the data. Note that I am using **4 physical CPU cores** to do this work. You need to use the number of physical cores available on your machine, although there is not sense in using more cores than you have taxa (in this case).

```
# go to the directory containing our config file and data
cd uce-tutorial

# run illumiprocessor

illumiprocessor \
    --input raw-fastq/ \
```

(continues on next page)

(continued from previous page)

```
    --output clean-fastq \
    --config illumiprocessor.conf \
    --cores 4
```

The output should look like the following:

```
2021-02-22 14:59:26,488 - illumiprocessor - INFO - ==================== Starting␣
↪illumiprocessor ===================
2021-02-22 14:59:26,489 - illumiprocessor - INFO - Version: 2.0.9
2021-02-22 14:59:26,489 - illumiprocessor - INFO - Argument --config: illumiprocessor.
↪conf
2021-02-22 14:59:26,489 - illumiprocessor - INFO - Argument --cores: 4
2021-02-22 14:59:26,489 - illumiprocessor - INFO - Argument --input: /scratch/
↪bfaircloth/uce-tutorial/raw-fastq
2021-02-22 14:59:26,490 - illumiprocessor - INFO - Argument --log_path: None
2021-02-22 14:59:26,490 - illumiprocessor - INFO - Argument --min_len: 40
2021-02-22 14:59:26,490 - illumiprocessor - INFO - Argument --no_merge: False
2021-02-22 14:59:26,490 - illumiprocessor - INFO - Argument --output: /scratch/
↪bfaircloth/uce-tutorial/clean-fastq
2021-02-22 14:59:26,490 - illumiprocessor - INFO - Argument --phred: phred33
2021-02-22 14:59:26,491 - illumiprocessor - INFO - Argument --r1_pattern: None
2021-02-22 14:59:26,491 - illumiprocessor - INFO - Argument --r2_pattern: None
2021-02-22 14:59:26,491 - illumiprocessor - INFO - Argument --se: False
2021-02-22 14:59:26,491 - illumiprocessor - INFO - Argument --trimmomatic: /home/bcf/
↪conda/envs/phyluce/bin/trimmomatic
2021-02-22 14:59:26,491 - illumiprocessor - INFO - Argument --verbosity: INFO
2021-02-22 14:59:26,904 - illumiprocessor - INFO - Trimming samples with Trimmomatic
Running....
2021-02-22 14:59:36,754 - illumiprocessor - INFO - =================== Completed␣
↪illumiprocessor ===================
```

Notice that the program has created a `log` file showing what it did, and it has also created a new directory holding the clean data that has the name `clean-fastq` (what you told it to name the directory). Within that new directory, there are taxon-specific folder for the cleaned reads. More specifically, your directory structure should look similar to the following (I've collapsed the list of raw-reads):

```
uce-tutorial
+── clean-fastq
│   +── alligator_mississippiensis
│   +── anolis_carolinensis
│   +── gallus_gallus
│   +── mus_musculus
+── illumiprocessor.conf
+── illumiprocessor.log
+── raw-fastq
```

Within each organism specific directory, there are more files and folders:

```
uce-tutorial
+── clean-fastq
│   +── alligator_mississippiensis
│   │   +── adapters.fasta
│   │   +── raw-reads
│   │   +── split-adapter-quality-trimmed
│   │   +── stats
│   +── anolis_carolinensis
```

(continues on next page)

(continued from previous page)

```
│       │   +── adapters.fasta
│       │   +── raw-reads
│       │   +── split-adapter-quality-trimmed
│       │   +── stats
│       +── gallus_gallus
│       │   +── adapters.fasta
│       │   +── raw-reads
│       │   +── split-adapter-quality-trimmed
│       │   +── stats
│       +── mus_musculus
│           +── adapters.fasta
│           +── raw-reads
│           +── split-adapter-quality-trimmed
│           +── stats
+── illumiprocessor.conf
+── illumiprocessor.log
+── raw-fastq
```

And, within each of those directories nested within the species-specific directory, there are additional files or links to files:

```
uce-tutorial
+── clean-fastq
│   +── alligator_mississippiensis
│   │   +── adapters.fasta
│   │   +── raw-reads
│   │   │   +── alligator_mississippiensis-READ1.fastq.gz -> <PATH>
│   │   │   +── alligator_mississippiensis-READ2.fastq.gz -> <PATH>
│   │   +── split-adapter-quality-trimmed
│   │   │   +── alligator_mississippiensis-READ1.fastq.gz
│   │   │   +── alligator_mississippiensis-READ2.fastq.gz
│   │   │   +── alligator_mississippiensis-READ-singleton.fastq.gz
│   │   +── stats
│   │       +── alligator_mississippiensis-adapter-contam.txt
│   +── anolis_carolinensis
│   +── gallus_gallus
│   +── mus_musculus
+── illumiprocessor.conf
+── illumiprocessor.log
+── raw-fastq
```

I have collapsed the listing to show only the first taxon.

The -> in the *raw-reads* directory above means there are symlinks to the files. I have removed the file paths and replaced them with *<PATH>* so that the figure will fit on a page.

The really important information is in the *split-adapter-quality-trimmed* directory - which now holds our reads that have had adapter-contamination and low-quality bases removed. Within this *split-adapter-quality-trimmed* directory, the *READ1* and *READ2* files hold reads that remain in a pair (the reads are in the same consecutive order in each file). The *READ-singleton* file holds READ1 reads **OR** READ2 reads that lost their "mate" or "paired-read" because of trimming or removal.

Quality control

You might want to get some idea of what effect the trimming has on read counts and overall read lengths. There are certainly other (better) tools out there to do this (like FastQC), but you can get a reasonable idea of how good your reads are by running the following, which will output a CSV listing of read stats by sample:

```
# move to the directory holding our cleaned reads
cd clean-fastq/

# run this script against all directories of reads

for i in *;
do
    phyluce_assembly_get_fastq_lengths --input $i/split-adapter-quality-trimmed/ --
↪csv;
done
```

The output you see should look like this:

```
All files in dir with alligator_mississippiensis-READ1.fastq.gz,93699,8418476,89.
↪84595353205476,0.059508742244529164,40,100,100.0
All files in dir with anolis_carolinensis-READ-singleton.fastq.gz,92184,8401336,91.
↪13659637247244,0.048890234925557836,40,100,100.0
All files in dir with gallus_gallus-READ1.fastq.gz,99444,21218771,213.37406982824504,
↪0.16122899415574637,40,251,250.0
All files in dir with mus_musculus-READ2.fastq.gz,89841,8165734,90.89095179261139,0.
↪052266485638855914,40,100,100.
```

Now, we're ready to assemble our reads.

Assemble the data

phyluce has several options for assembly - you can use velvet, abyss, or spades. For this tutorial, we are going to use spades, because it seems to works best for most purposes, it is easy to install and run, and it works consistently. The helper programs for the other assemblers use the same config file, so you can easily experiment with all of the assemblers.

To run an assembly, we need to create a another configuration file. The assembly configuration file looks like the following, assuming we want to assemble all of our data from the organisms above:

```
[samples]
alligator_mississippiensis:/path/to/the/uce-tutorial/clean-fastq/alligator_
↪mississippiensis/split-adapter-quality-trimmed/
anolis_carolinensis:/path/to/the/uce-tutorial/clean-fastq/anolis_carolinensis/split-
↪adapter-quality-trimmed/
gallus_gallus:/path/to/the/uce-tutorial/clean-fastq/gallus_gallus/split-adapter-
↪quality-trimmed/
mus_musculus:/scratch/bfaircloth-uce-tutorial/clean-fastq/mus_musculus/split-adapter-
↪quality-trimmed/
```

You need to modify this file to use the path to the clean read data **on your computer** (`/path/to/the/` is a placeholder, here). You will save this into a file named `assembly.conf` at the top of our `uce-tutorial` directory:

```
uce-tutorial
+── assembly.conf
+── clean-fastq
+── illumiprocessor.conf
+── illumiprocessor.log
+── phyluce_assembly_assemblo_trinity.log
+── raw-fastq
+── trinity-assemblies
```

If you want to change the names on the left hand side of the colon in the config file, you can do so, but the paths on the right hand side need to point to our "clean" UCE raw reads. If you have files in multiple locations, you can use any number of different paths on the right-hand side of the colon.

Attention: Although you can easily input new PATHs in this file, the **structure** of the data below the PATH you use must be the same - meaning that the structure and naming scheme for READ1, READ2, and READ-singleton must be the same. Or, put another way, the assembly programs for phyluce assume the data **always** look like the following:

```
<some working folder name>
+── clean-fastq
    +── alligator_mississippiensis
    │     +── split-adapter-quality-trimmed
    │        +── alligator_mississippiensis-READ1.fastq.gz
    │        +── alligator_mississippiensis-READ2.fastq.gz
    │        +── alligator_mississippiensis-READ-singleton.fastq.gz
    +── anolis_carolinensis
        +── split-adapter-quality-trimmed
            +── anolis_carolinensis-READ1.fastq.gz
            +── anolis_carolinensis-READ2.fastq.gz
            +── anolis_carolinensis-READ-singleton.fastq.gz
```

Now that we have that file created, copy it to our working directory, and run the `phyluce_assembly_assemblo_spades` program:

```
# make sure we are at the top-level of our uce tutorial directory
cd uce-tutorial

# run the assembly
phyluce_assembly_assemblo_spades \
    --conf assembly.conf \
    --output spades-assemblies \
    --cores 12
```

Warning: Note that I am using 12 physical CPU cores to do this work. You need to use the number of physical cores available on **your** machine. `phyluce_assembly_assemblo_spades` assumes you have **at least** 8 GB of RAM on your system, and it is better to have much more. If you use more CPU cores than you have or you specify more RAM than you have, the job can fail.

You can adjust the RAM dedicated to the job using the `--memory` option, which takes an integer value (in GB RAM).

Note: If you are wondering, Trinity is no longer supported in phyluce

As the assembly proceeds, you should see output similar to the following:

```
2021-02-26 21:12:16,615 - phyluce_assembly_assemblo_spades - INFO - ===========␣
↪Starting phyluce_assembly_assemblo_spades ===========
2021-02-26 21:12:16,615 - phyluce_assembly_assemblo_spades - INFO - Version: 1.7.0
2021-02-26 21:12:16,615 - phyluce_assembly_assemblo_spades - INFO - Commit: None
2021-02-26 21:12:16,615 - phyluce_assembly_assemblo_spades - INFO - Argument --
↪config: /data/assembly.conf
```

(continues on next page)

(continued from previous page)

```
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --cores:␣
↪12
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --dir:␣
↪None
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --do_not_
↪clean: False
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --log_
↪path: None
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --
↪memory: 8
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --
↪output: /data/spades-assemblies
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --
↪subfolder:
2021-02-26 21:12:16,616 - phyluce_assembly_assemblo_spades - INFO - Argument --
↪verbosity: INFO
2021-02-26 21:12:16,617 - phyluce_assembly_assemblo_spades - INFO - Getting input␣
↪filenames and creating output directories
2021-02-26 21:12:16,765 - phyluce_assembly_assemblo_spades - INFO - -------------␣
↪Processing alligator_mississippiensis -------------
2021-02-26 21:12:16,775 - phyluce_assembly_assemblo_spades - INFO - Finding fastq/
↪fasta files
2021-02-26 21:12:16,787 - phyluce_assembly_assemblo_spades - INFO - File type is fastq
2021-02-26 21:12:16,787 - phyluce_assembly_assemblo_spades - INFO - Running SPAdes␣
↪for PE data
2021-02-26 21:13:35,643 - phyluce_assembly_assemblo_spades - INFO - Symlinking␣
↪assembled contigs into /data/spades-assemblies/contigs
...[continued]...
2021-02-26 21:19:06,618 - phyluce_assembly_assemblo_spades - INFO - Symlinking␣
↪assembled contigs into /data/spades-assemblies/contigs
2021-02-26 21:19:06,624 - phyluce_assembly_assemblo_spades - INFO - ===========␣
↪Completed phyluce_assembly_assemblo_spades ==========
```

One the assembly is finished, have a look at the directory structure:

```
uce-tutorial
+── assembly.conf
+── clean-fastq
+── illumiprocessor.conf
+── illumiprocessor.log
+── phyluce_assembly_assemblo_trinity.log
+── raw-fastq
+── spades-assemblies
    +── alligator_mississippiensis_trinity
    │   +── contigs.fasta
    │   +── scaffolds.fasta
    │   +── spades.log
    +── anolis_carolinensis_trinity
    │   +── contigs.fasta
    │   +── scaffolds.fasta
    │   +── spades.log
    +── contigs
    │   +── alligator_mississippiensis.contigs.fasta -> ../alligator_mississippiensis_
↪trinity/contigs.fasta
    │   +── anolis_carolinensis.contigs.fasta -> ../anolis_carolinensis_trinity/
↪contigs.fasta
    │   +── gallus_gallus.contigs.fasta -> ../gallus_gallus_trinity/contigs.fasta
```

(continues on next page)

(continued from previous page)

```
    |    +-- mus_musculus.contigs.fasta -> ../mus_musculus_trinity/contigs.fasta
    +-- gallus_gallus_trinity
    |    +-- contigs.fasta
    |    +-- scaffolds.fasta
    |    +-- spades.log
    +-- mus_musculus_trinity
         +-- contigs.fasta
         +-- scaffolds.fasta
         +-- spades.log
```

Your species-specific assembly files are in the *spades-assemblies* directory nested within species-specific directories that correspond to the name you used in the *assembly.conf* file (to the left of the colon).

There is also a *contigs* directory within this folder. The *contigs* directory is the important one, because it contains symlinks to all of the species- specific contigs. This means that you can treat this single folder as if it contains all of your assembled contigs.

Assembly QC

We can get a sense of how well the assembly worked by running the following from the top of our working directory:

```
# run this script against all directories of reads

for i in spades-assemblies/contigs/*.fasta;
do
    phyluce_assembly_get_fasta_lengths --input $i --csv;
done
```

This should output something similar to the following. I've added the header as a comment:

```
# samples,contigs,total bp,mean length,95 CI length,min length,max length,median␣
↪legnth,contigs >1kb
alligator_mississippiensis.contigs.fasta,870,220436,253.37471264367815,6.
↪9967833874072225,56,3831,236.0,5
anolis_carolinensis.contigs.fasta,1136,355837,313.2367957746479,8.00195622090676,56,
↪3182,243.0,9
gallus_gallus.contigs.fasta,5228,2841631,543.5407421576128,2.8905564028218618,56,4117,
↪495.5,107
mus_musculus.contigs.fasta,1395,324498,232.61505376344087,6.895305881534584,56,1646,
↪88.0,13
```

Question: Why are my numbers slightly different than your numbers?

The process of read assembly often differs by operating system and sometimes by OS version, and some of these differences are due to libraries that underlie many of the assembly programs. Expect to see differences. You should not expect for them to be huge.

Attention: If you see max-contig sizes around 16KB (for vertebrates), that is commonly the entire or almost-entire mtDNA genome. You do not tend to see entire mtDNA assemblies when the input DNA was extracted from a source having few mitochondria (e.g. blood).

There are many, many other assembly QC steps you can run other than simply looking at the stats of the assembled contigs. We will not go into those here.

Finding UCE loci

Now that we've assembled our contigs from raw reads, it's time to find those contigs which are UCE loci and move aside those that are not. The directory structure before we do this should look like the following:

```
uce-tutorial
+-- assembly.conf
+-- clean-fastq
+-- illumiprocessor.conf
+-- illumiprocessor.log
+-- phyluce_assembly_assemblo_trinity.log
+-- raw-fastq
+-- trinity-assemblies
```

Before we locate UCE loci, you need to get the probe set used for the enrichments:

```
wget https://raw.githubusercontent.com/faircloth-lab/uce-probe-sets/master/uce-5k-
↪probe-set/uce-5k-probes.fasta
```

Now, our directory structure looks like:

```
uce-tutorial
+-- assembly.conf
+-- clean-fastq
+-- illumiprocessor.conf
+-- illumiprocessor.log
+-- phyluce_assembly_assemblo_trinity.log
+-- raw-fastq
+-- trinity-assemblies
+-- uce-5k-probes.fasta
```

Now, run the *phyluce_assembly_match_contigs_to_probes* program:

```
phyluce_assembly_match_contigs_to_probes \
    --contigs spades-assemblies/contigs \
    --probes uce-5k-probes.fasta \
    --output uce-search-results
```

You should see output similar to the following (also stored in *phyluce_assembly_assemblo_trinity.log*):

```
2021-02-26 21:37:43,108 - phyluce_assembly_match_contigs_to_probes - INFO - =======␣
↪Starting phyluce_assembly_match_contigs_to_probes =======
2021-02-26 21:37:43,108 - phyluce_assembly_match_contigs_to_probes - INFO - Version:␣
↪1.7.0
2021-02-26 21:37:43,109 - phyluce_assembly_match_contigs_to_probes - INFO - Commit:␣
↪None
2021-02-26 21:37:43,109 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-contigs: /data/spades-assemblies/contigs
2021-02-26 21:37:43,109 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-csv: test-output.csv
2021-02-26 21:37:43,109 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-dupefile: None
2021-02-26 21:37:43,110 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-keep_duplicates: None
```

(continues on next page)

(continued from previous page)

```
2021-02-26 21:37:43,110 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-log_path: None
2021-02-26 21:37:43,110 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-min_coverage: 80
2021-02-26 21:37:43,110 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-min_identity: 80
2021-02-26 21:37:43,110 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-output: /data/uce-search-results
2021-02-26 21:37:43,111 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-probes: /data/uce-5k-probes.fasta
2021-02-26 21:37:43,111 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-regex: ^(uce-\d+)(?:_p\d+.*)
2021-02-26 21:37:43,111 - phyluce_assembly_match_contigs_to_probes - INFO - Argument -
↪-verbosity: INFO
2021-02-26 21:37:43,225 - phyluce_assembly_match_contigs_to_probes - INFO - Creating␣
↪the UCE-match database
2021-02-26 21:37:43,266 - phyluce_assembly_match_contigs_to_probes - INFO -␣
↪Processing contig data
2021-02-26 21:37:43,266 - phyluce_assembly_match_contigs_to_probes - INFO - ----------
↪-------------------------------------------------------
2021-02-26 21:37:44,680 - phyluce_assembly_match_contigs_to_probes - INFO - alligator_
↪mississippiensis: 422 (48.51%) uniques of 870 contigs, 0 dupe probe matches, 3 UCE␣
↪loci removed for matching multiple contigs, 2 contigs removed for matching multiple␣
↪UCE loci
2021-02-26 21:37:45,940 - phyluce_assembly_match_contigs_to_probes - INFO - anolis_
↪carolinensis: 399 (35.12%) uniques of 1136 contigs, 0 dupe probe matches, 0 UCE␣
↪loci removed for matching multiple contigs, 2 contigs removed for matching multiple␣
↪UCE loci
2021-02-26 21:37:53,686 - phyluce_assembly_match_contigs_to_probes - INFO - gallus_
↪gallus: 3492 (66.79%) uniques of 5228 contigs, 0 dupe probe matches, 22 UCE loci␣
↪removed for matching multiple contigs, 37 contigs removed for matching multiple UCE␣
↪loci
2021-02-26 21:37:54,853 - phyluce_assembly_match_contigs_to_probes - INFO - mus_
↪musculus: 324 (23.23%) uniques of 1395 contigs, 0 dupe probe matches, 0 UCE loci␣
↪removed for matching multiple contigs, 1 contigs removed for matching multiple UCE␣
↪loci
2021-02-26 21:37:54,853 - phyluce_assembly_match_contigs_to_probes - INFO - ----------
↪-------------------------------------------------------
2021-02-26 21:37:54,853 - phyluce_assembly_match_contigs_to_probes - INFO - The LASTZ␣
↪alignments are in /data/uce-search-results
2021-02-26 21:37:54,854 - phyluce_assembly_match_contigs_to_probes - INFO - The UCE␣
↪match database is in /data/uce-search-results/probe.matches.sqlite
2021-02-26 21:37:54,854 - phyluce_assembly_match_contigs_to_probes - INFO - =======␣
↪Completed phyluce_assembly_match_contigs_to_probes ======
```

The header info at the top tells us exactly what version of the code we are running and keeps track of our options. The important output is:

```
alligator_mississippiensis: 422 (48.51%) uniques of 870 contigs, 0 dupe probe matches,
↪ 3 UCE loci removed for matching multiple contigs, 2 contigs removed for matching␣
↪multiple UCE loci
anolis_carolinensis: 399 (35.12%) uniques of 1136 contigs, 0 dupe probe matches, 0␣
↪UCE loci removed for matching multiple contigs, 2 contigs removed for matching␣
↪multiple UCE loci
gallus_gallus: 3492 (66.79%) uniques of 5228 contigs, 0 dupe probe matches, 22 UCE␣
↪loci removed for matching multiple contigs, 37 contigs removed for matching␣
↪multiple UCE loci
```

(continues on next page)

(continued from previous page)

```
mus_musculus: 324 (23.23%) uniques of 1395 contigs, 0 dupe probe matches, 0 UCE loci␣
↪removed for matching multiple contigs, 1 contigs removed for matching multiple UCE␣
↪loci
```

Which we can break down to the following (for *alligator_mississippiensis*):

```
alligator_mississippiensis:
    422 (48.51%) uniques of 870 contigs
    0 dupe probe matches
    3 UCE loci removed for matching multiple contigs
    2 contigs removed for matching multiple UCE loci
```

These are the capture data for the *alligator_mississippiensis* sample. We targeted 5k UCE loci in this sample and recovered roughly 422 of those loci **in this subsampled set of reads**. Before reaching that total of 422 loci, we removed 3 UCE loci and 2 contigs from the data set because they looked like duplicates (probes supposedly targeting different loci hit the same contig or two supposedly different contigs hit probes designed for a single UCE locus).

Question: Why is the count of UCE loci different by sample?

For these example data, we enriched some samples (alligator_mississippiensis and gallus_gallus) for 5k UCE loci, while we enriched others (anolis_carolinensis and mus_musculus) for 2.5k UCE loci. Additionally, the 2.5k UCE enrichments did not work very well (operator error). Finally, because we have subsampled the data to make the files of reasonable size for the tutorial, that process removes lots of read that **would** have assembled into UCE contigs (e.g., if we use ALL the data, we recover 4000+ UCE contigs for alligator).

The directory structure now looks like the following (everything collapsed but the *uce-search-results directory*):

```
uce-tutorial
+-- assembly.conf
+-- clean-fastq
+-- illumiprocessor.conf
+-- illumiprocessor.log
+-- phyluce_assembly_assemblo_trinity.log
+-- phyluce_assembly_match_contigs_to_probes.log
+-- raw-fastq
+-- trinity-assemblies
+-- uce-5k-probes.fasta
+-- uce-search-results
    +-- alligator_mississippiensis.contigs.lastz
    +-- anolis_carolinensis.contigs.lastz
    +-- gallus_gallus.contigs.lastz
    +-- mus_musculus.contigs.lastz
    +-- probe.matches.sqlite
```

The search we just ran created lastz search result files for each taxon, and stored summary results of these searches in the *probe.matches.sqlite* database (see *The probe.matches.sqlite database* for more information on this database and its structure).

Extracting UCE loci

Now that we have located UCE loci, we need to determine which taxa we want in our analysis, create a list of those taxa, and then generate a list of which UCE loci we enriched in each taxon (the "data matrix configuration file"). We will then use this list to extract FASTA data for each taxon for each UCE locus.

First, we need to decide which taxa we want in our "taxon set". So, we create a configuration file like so:

```
[all]
alligator_mississippiensis
anolis_carolinensis
gallus_gallus
mus_musculus
```

These names need to match the assembly names we used. Here, we have just put all 4 taxa in a list that we named *all*. However, we can adjust this list in many ways (see *Creating a data matrix configuration file*).

Save this file as *taxon-set.conf* at the top level of our *uce-tutorial* directory. The directory should look like this, now:

```
uce-tutorial
+—— assembly.conf
+—— clean-fastq
+—— illumiprocessor.conf
+—— illumiprocessor.log
+—— phyluce_assembly_assemblo_trinity.log
+—— phyluce_assembly_match_contigs_to_probes.log
+—— raw-fastq
+—— taxon-set.conf
+—— trinity-assemblies
+—— uce-5k-probes.fasta
+—— uce-search-results
```

Now that we have this file created, we do the following to create the initial list of loci for each taxon:

```
# create an output directory for this taxon set - this just keeps
# things neat
cd uce-tutorial
mkdir -p taxon-sets/all

# create the data matrix configuration file
phyluce_assembly_get_match_counts \
    --locus-db uce-search-results/probe.matches.sqlite \
    --taxon-list-config taxon-set.conf \
    --taxon-group 'all' \
    --incomplete-matrix \
    --output taxon-sets/all/all-taxa-incomplete.conf
```

The output should look like the following:

```
2021-03-01 15:46:14,277 - phyluce_assembly_get_match_counts - INFO - ===========␣
↪Starting phyluce_assembly_get_match_counts ==========
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Version: 1.7.0
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Commit: None
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Argument --
↪extend_locus_db: None
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Argument --
↪incomplete_matrix: True
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Argument --keep_
↪counts: False
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Argument --locus_
↪db: /data/uce-search-results/probe.matches.sqlite
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Argument --log_
↪path: None
2021-03-01 15:46:14,278 - phyluce_assembly_get_match_counts - INFO - Argument --
↪optimize: False
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --
↪output: /data/taxon-sets/all/all-taxa-incomplete.conf
```

(continues on next page)

(continued from previous page)

```
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --
↪random: False
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --
↪sample_size: 10
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --
↪samples: 10
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --
↪silent: False
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --taxon_
↪group: all
2021-03-01 15:46:14,279 - phyluce_assembly_get_match_counts - INFO - Argument --taxon_
↪list_config: /data/taxon-set.conf
2021-03-01 15:46:14,280 - phyluce_assembly_get_match_counts - INFO - Argument --
↪verbosity: INFO
2021-03-01 15:46:14,319 - phyluce_assembly_get_match_counts - INFO - There are 4 taxa␣
↪in the taxon-group '[all]' in the config file taxon-set.conf
2021-03-01 15:46:14,319 - phyluce_assembly_get_match_counts - INFO - Getting UCE␣
↪names from database
2021-03-01 15:46:44,488 - phyluce_assembly_get_match_counts - INFO - There are 5041␣
↪total UCE loci in the database
2021-03-01 15:46:44,567 - phyluce_assembly_get_match_counts - INFO - Getting UCE␣
↪matches by organism to generate a INCOMPLETE matrix
2021-03-01 15:46:44,569 - phyluce_assembly_get_match_counts - INFO - There are 3653␣
↪UCE loci in an INCOMPLETE matrix
2021-03-01 15:46:44,569 - phyluce_assembly_get_match_counts - INFO - Writing the taxa␣
↪and loci in the data matrix to /data/taxon-sets/all/all-taxa-incomplete.conf
2021-03-01 15:46:44,574 - phyluce_assembly_get_match_counts - INFO - ==========␣
↪Completed phyluce_assembly_get_match_counts ==========
```

And, our directory structure should now look like this (collapsing all but *taxon-sets*):

```
uce-tutorial
+-- assembly.conf
+-- clean-fastq
+-- illumiprocessor.conf
+-- illumiprocessor.log
+-- phyluce_assembly_assemblo_trinity.log
+-- phyluce_assembly_get_match_counts.log
+-- phyluce_assembly_match_contigs_to_probes.log
+-- taxon-set.conf
+-- taxon-sets
|   +-- all
|       +-- all-taxa-incomplete.conf
+-- trinity-assemblies
+-- uce-5k-probes.fasta
+-- uce-search-results
```

Now, we need to extract FASTA data that correspond to the loci in *all-taxa-incomplete.conf*:

```
# change to the taxon-sets/all directory
cd taxon-sets/all

# make a log directory to hold our log files - this keeps things neat
mkdir log

# get FASTA data for taxa in our taxon set
phyluce_assembly_get_fastas_from_match_counts \
```

(continues on next page)

(continued from previous page)

```
    --contigs ../../spades-assemblies/contigs \
    --locus-db ../../uce-search-results/probe.matches.sqlite \
    --match-count-output all-taxa-incomplete.conf \
    --output all-taxa-incomplete.fasta \
    --incomplete-matrix all-taxa-incomplete.incomplete \
    --log-path log
```

The output should look something like the following:

```
2021-03-01 15:47:55,574 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪===== Starting phyluce_assembly_get_fastas_from_match_counts ====
2021-03-01 15:47:55,575 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Version: 1.7.0
2021-03-01 15:47:55,575 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Commit: None
2021-03-01 15:47:55,575 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --contigs: /data/spades-assemblies/contigs
2021-03-01 15:47:55,575 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --extend_locus_contigs: None
2021-03-01 15:47:55,575 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --extend_locus_db: None
2021-03-01 15:47:55,575 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --incomplete_matrix: /data/taxon-sets/all/all-taxa-incomplete.incomplete
2021-03-01 15:47:55,576 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --locus_db: /data/uce-search-results/probe.matches.sqlite
2021-03-01 15:47:55,576 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --log_path: /data/taxon-sets/all/log
2021-03-01 15:47:55,576 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --match_count_output: /data/taxon-sets/all/all-taxa-incomplete.conf
2021-03-01 15:47:55,576 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --output: /data/taxon-sets/all/all-taxa-incomplete.fasta
2021-03-01 15:47:55,576 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Argument --verbosity: INFO
2021-03-01 15:47:55,609 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪There are 4 taxa in the match-count-config file named all-taxa-incomplete.conf
2021-03-01 15:47:55,612 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪There are 3653 UCE loci in an INCOMPLETE matrix
2021-03-01 15:47:55,614 - phyluce_assembly_get_fastas_from_match_counts - INFO - -----
↪----Getting UCE loci for alligator_mississippiensis---------
2021-03-01 15:47:55,925 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪There are 422 UCE loci for alligator_mississippiensis
2021-03-01 15:47:55,925 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Parsing and renaming contigs for alligator_mississippiensis
2021-03-01 15:47:56,396 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Writing missing locus information to /data/taxon-sets/all/all-taxa-incomplete.
↪incomplete
2021-03-01 15:47:56,399 - phyluce_assembly_get_fastas_from_match_counts - INFO - -----
↪--------Getting UCE loci for anolis_carolinensis------------
2021-03-01 15:47:56,638 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪There are 399 UCE loci for anolis_carolinensis
2021-03-01 15:47:56,638 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Parsing and renaming contigs for anolis_carolinensis
2021-03-01 15:47:57,077 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Replaced <20 ambiguous bases (N) in 7 contigs for anolis_carolinensis
2021-03-01 15:47:57,077 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Writing missing locus information to /data/taxon-sets/all/all-taxa-incomplete.
↪incomplete
```

(continues on next page)

(continued from previous page)

```
2021-03-01 15:47:57,079 - phyluce_assembly_get_fastas_from_match_counts - INFO - -----
↪-----------Getting UCE loci for gallus_gallus---------------
2021-03-01 15:47:58,940 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪There are 3492 UCE loci for gallus_gallus
2021-03-01 15:47:58,940 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Parsing and renaming contigs for gallus_gallus
2021-03-01 15:48:01,965 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Writing missing locus information to /data/taxon-sets/all/all-taxa-incomplete.
↪incomplete
2021-03-01 15:48:01,966 - phyluce_assembly_get_fastas_from_match_counts - INFO - -----
↪-----------Getting UCE loci for mus_musculus----------------
2021-03-01 15:48:02,168 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪There are 324 UCE loci for mus_musculus
2021-03-01 15:48:02,168 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Parsing and renaming contigs for mus_musculus
2021-03-01 15:48:02,508 - phyluce_assembly_get_fastas_from_match_counts - INFO -␣
↪Writing missing locus information to /data/taxon-sets/all/all-taxa-incomplete.
↪incomplete
2021-03-01 15:48:02,519 - phyluce_assembly_get_fastas_from_match_counts - INFO - ====␣
↪Completed phyluce_assembly_get_fastas_from_match_counts ====
```

And, our directory structure should now look like this (collapsing all but *taxon-sets*):

```
uce-tutorial
+-- assembly.conf
+-- clean-fastq
+-- illumiprocessor.conf
+-- illumiprocessor.log
+-- phyluce_assembly_assemblo_trinity.log
+-- phyluce_assembly_get_match_counts.log
+-- phyluce_assembly_match_contigs_to_probes.log
+-- taxon-set.conf
+-- taxon-sets
|   +-- all
|       +-- all-taxa-incomplete.conf
|       +-- all-taxa-incomplete.fasta
|       +-- all-taxa-incomplete.incomplete
|       +-- log
|           +-- phyluce_assembly_get_fastas_from_match_counts.log
+-- phyluce_assembly_get_fastas_from_match_counts.log
+-- trinity-assemblies
+-- uce-5k-probes.fasta
+-- uce-search-results
```

The extracted FASTA data are in a monolithic FASTA file (all data for all organisms) named *all-taxa-incomplete.fasta*.

Exploding the monolithic FASTA file

Lots of times we want to know individual statistics on UCE assemblies for a given taxon. We can do that by exploding the monolithic fasta file into a file of UCE loci that we have enriched by taxon, then running stats on those exploded files. To do that, run the following:

```
# explode the monolithic FASTA by taxon (you can also do by locus)
phyluce_assembly_explode_get_fastas_file \
    --input all-taxa-incomplete.fasta \
```

(continues on next page)

(continued from previous page)

```
    --output exploded-fastas \
    --by-taxon

# get summary stats on the FASTAS
for i in exploded-fastas/*.fasta;
do
    phyluce_assembly_get_fasta_lengths --input $i --csv;
done

# samples,contigs,total bp,mean length,95 CI length,min length,max length,median␣
↪legnth,contigs >1kb
alligator-mississippiensis.unaligned.fasta,422,118864,281.66824644549763,9.
↪03914154267783,206,3831,250.5,1
anolis-carolinensis.unaligned.fasta,399,206926,518.6115288220551,11.261692813904311,
↪206,1115,525.0,1
gallus-gallus.unaligned.fasta,3492,2157267,617.7740549828179,3.075870593111506,307,
↪1503,607.0,81
mus-musculus.unaligned.fasta,324,188082,580.5,13.930004844688803,207,1058,623.5,6
```

Aligning UCE loci

You have lots of options when aligning UCE loci. You can align the loci and use those alignments with no trimming, you can edge-trim the alignments following some algorithm, and you can end+internally trim alignments following some algorithm. It's hard to say what is best in all situations. When taxa are "closely" related (< 30-50 MYA, perhaps), I think that edge-trimming alignments is reasonable. When the taxa you are interested in span a wider range of divergence times (> 50 MYA), you may want to think about internal trimming.

How you accomplish you edge- or internal-trimming is also a decision you need to make. In phyluce, we implement our edge-trimming algorithm by running the alignment program "as-is" (i.e., without the *–no-trim*) option. We do internal-trimming by turning off trimming using *–no-trim*, then passing the resulting alignments (in FASTA format) to a parallel wrapper around Gblocks.

You also have a choice of aligner - mafft or muscle (or you can externally align UCE loci using a tool like SATé, as well).

Generally, I would use mafft.

Edge trimming

Edge trimming your alignments is a relatively simple matter. You can run edge trimming, as follows:

```
# make sure we are in the correct directory
cd uce-tutorial/taxon-sets/all

# align the data
phyluce_align_seqcap_align \
    --input all-taxa-incomplete.fasta \
    --output mafft-nexus-edge-trimmed \
    --taxa 4 \
    --aligner mafft \
    --cores 12 \
    --incomplete-matrix \
    --log-path log
```

Warning: Note that I am using 12 physical CPU cores here. You need to use the number of physical cores available on *your* machine.

The output should look like this:

```
2021-03-01 15:51:21,854 - phyluce_align_seqcap_align - INFO - ============== Starting␣
↪phyluce_align_seqcap_align ==============
2021-03-01 15:51:21,855 - phyluce_align_seqcap_align - INFO - Version: 1.7.0
2021-03-01 15:51:21,855 - phyluce_align_seqcap_align - INFO - Commit: None
2021-03-01 15:51:21,856 - phyluce_align_seqcap_align - INFO - Argument --aligner:␣
↪mafft
2021-03-01 15:51:21,856 - phyluce_align_seqcap_align - INFO - Argument --ambiguous:␣
↪False
2021-03-01 15:51:21,856 - phyluce_align_seqcap_align - INFO - Argument --cores: 12
2021-03-01 15:51:21,856 - phyluce_align_seqcap_align - INFO - Argument --input: /data/
↪taxon-sets/all/all-taxa-incomplete.fasta
2021-03-01 15:51:21,857 - phyluce_align_seqcap_align - INFO - Argument --log_path: /
↪data/taxon-sets/all/log
2021-03-01 15:51:21,857 - phyluce_align_seqcap_align - INFO - Argument --max_
↪divergence: 0.2
2021-03-01 15:51:21,857 - phyluce_align_seqcap_align - INFO - Argument --min_length:␣
↪100
2021-03-01 15:51:21,857 - phyluce_align_seqcap_align - INFO - Argument --no_trim:␣
↪False
2021-03-01 15:51:21,857 - phyluce_align_seqcap_align - INFO - Argument --notstrict:␣
↪True
2021-03-01 15:51:21,858 - phyluce_align_seqcap_align - INFO - Argument --output: /
↪data/taxon-sets/all/mafft-nexus-edge-trimmed
2021-03-01 15:51:21,858 - phyluce_align_seqcap_align - INFO - Argument --output_
↪format: nexus
2021-03-01 15:51:21,858 - phyluce_align_seqcap_align - INFO - Argument --proportion:␣
↪0.65
2021-03-01 15:51:21,858 - phyluce_align_seqcap_align - INFO - Argument --taxa: 4
2021-03-01 15:51:21,858 - phyluce_align_seqcap_align - INFO - Argument --threshold: 0.
↪65
2021-03-01 15:51:21,859 - phyluce_align_seqcap_align - INFO - Argument --verbosity:␣
↪INFO
2021-03-01 15:51:21,859 - phyluce_align_seqcap_align - INFO - Argument --window: 20
2021-03-01 15:51:21,859 - phyluce_align_seqcap_align - INFO - Building the locus␣
↪dictionary
2021-03-01 15:51:21,859 - phyluce_align_seqcap_align - INFO - Removing ALL sequences␣
↪with ambiguous bases...
2021-03-01 15:51:22,328 - phyluce_align_seqcap_align - WARNING - DROPPED locus uce-
↪4698. Too few taxa (N < 3).
[many more loci dropped here]
2021-03-01 15:51:22,750 - phyluce_align_seqcap_align - INFO - Aligning with MAFFT
2021-03-01 15:51:22,751 - phyluce_align_seqcap_align - INFO - Alignment begins. 'X'␣
↪indicates dropped alignments (these are reported after alignment)
.................[continued]
2021-03-01 15:51:28,544 - phyluce_align_seqcap_align - INFO - Alignment ends
2021-03-01 15:51:28,545 - phyluce_align_seqcap_align - INFO - Writing output files
2021-03-01 15:51:28,788 - phyluce_align_seqcap_align - INFO - ==============␣
↪Completed phyluce_align_seqcap_align =============
```

The . values that you see represent loci that were aligned and succesfully trimmed. Any *X* values that you see represent loci that were removed because trimming reduced their length to effectively nothing.

Attention: The number of potential alignments dropped here is abnormally large becase our **sample size is so small (n=4)**.

The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+── assembly.conf
...
+── taxon-sets
│       +── all
│           +── all-taxa-incomplete.conf
│           +── all-taxa-incomplete.fasta
│           +── all-taxa-incomplete.incomplete
│           +── exploded-fastas
│           +── log
│           +── mafft-nexus-edge-trimmed
│               +── uce-1008.nexus
│               +── uce-1014.nexus
│               +── uce-1039.nexus
│               ...
│               +── uce-991.nexus
...
+── uce-search-results
```

We can output summary stats for these alignments by running the following program:

```
phyluce_align_get_align_summary_data \
    --alignments mafft-nexus-edge-trimmed \
    --cores 12 \
    --log-path log
```

Warning: Note that I am using 12 physical CPU cores here. You need to use the number of physical cores available on *your* machine.

The output from the program should look like:

```
2021-03-01 15:58:43,241 - phyluce_align_get_align_summary_data - INFO - =========␣
↪Starting phyluce_align_get_align_summary_data =========
2021-03-01 15:58:43,241 - phyluce_align_get_align_summary_data - INFO - Version: 1.7.0
2021-03-01 15:58:43,241 - phyluce_align_get_align_summary_data - INFO - Commit: None
2021-03-01 15:58:43,241 - phyluce_align_get_align_summary_data - INFO - Argument --
↪alignments: /data/taxon-sets/all/mafft-nexus-edge-trimmed
2021-03-01 15:58:43,241 - phyluce_align_get_align_summary_data - INFO - Argument --
↪cores: 12
2021-03-01 15:58:43,241 - phyluce_align_get_align_summary_data - INFO - Argument --
↪input_format: nexus
2021-03-01 15:58:43,242 - phyluce_align_get_align_summary_data - INFO - Argument --
↪log_path: /data/taxon-sets/all/log
2021-03-01 15:58:43,242 - phyluce_align_get_align_summary_data - INFO - Argument --
↪output: None
2021-03-01 15:58:43,242 - phyluce_align_get_align_summary_data - INFO - Argument --
↪show_taxon_counts: False
2021-03-01 15:58:43,242 - phyluce_align_get_align_summary_data - INFO - Argument --
↪verbosity: INFO
```

(continues on next page)

(continued from previous page)

```
2021-03-01 15:58:43,242 - phyluce_align_get_align_summary_data - INFO - Getting␣
↪alignment files
2021-03-01 15:58:43,251 - phyluce_align_get_align_summary_data - INFO - Computing␣
↪summary statistics using 12 cores
2021-03-01 15:58:43,596 - phyluce_align_get_align_summary_data - INFO - --------------
↪--------- Alignment summary -----------------------
2021-03-01 15:58:43,597 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪loci:  190
2021-03-01 15:58:43,597 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪length:        86,032
2021-03-01 15:58:43,597 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪mean:  452.80
2021-03-01 15:58:43,597 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪95% CI:        21.51
2021-03-01 15:58:43,597 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪min:   126
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪max:   907
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - --------------
↪----- Informative Sites summary -------------------
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - [Sites] loci:␣
↪       190
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - [Sites]␣
↪total:      58
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - [Sites] mean:␣
↪       0.31
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - [Sites] 95%␣
↪CI:     0.18
2021-03-01 15:58:43,598 - phyluce_align_get_align_summary_data - INFO - [Sites] min: ␣
↪       0
2021-03-01 15:58:43,599 - phyluce_align_get_align_summary_data - INFO - [Sites] max: ␣
↪       12
2021-03-01 15:58:43,600 - phyluce_align_get_align_summary_data - INFO - --------------
↪----------- Taxon summary -------------------------
2021-03-01 15:58:43,600 - phyluce_align_get_align_summary_data - INFO - [Taxa] mean: ␣
↪       3.15
2021-03-01 15:58:43,600 - phyluce_align_get_align_summary_data - INFO - [Taxa] 95%␣
↪CI:      0.05
2021-03-01 15:58:43,600 - phyluce_align_get_align_summary_data - INFO - [Taxa] min:  ␣
↪       3
2021-03-01 15:58:43,600 - phyluce_align_get_align_summary_data - INFO - [Taxa] max:  ␣
↪       4
2021-03-01 15:58:43,601 - phyluce_align_get_align_summary_data - INFO - --------------
↪--- Missing data from trim summary ----------------
2021-03-01 15:58:43,601 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪mean:          9.36
2021-03-01 15:58:43,601 - phyluce_align_get_align_summary_data - INFO - [Missing] 95%␣
↪CI:    0.84
2021-03-01 15:58:43,601 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪min:           0.00
2021-03-01 15:58:43,601 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪max:           32.67
2021-03-01 15:58:43,603 - phyluce_align_get_align_summary_data - INFO - --------------
↪------ Character count summary --------------------
2021-03-01 15:58:43,603 - phyluce_align_get_align_summary_data - INFO - [All␣
↪characters]    268,288
2021-03-01 15:58:43,603 - phyluce_align_get_align_summary_data - INFO - [Nucleotides]␣
↪               235,385
```

(continues on next page)

(continued from previous page)

```
2021-03-01 15:58:43,603 - phyluce_align_get_align_summary_data - INFO - --------------
↪-- Data matrix completeness summary ---------------
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 50%] ␣
↪               190 alignments
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 55%] ␣
↪               190 alignments
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 60%] ␣
↪               190 alignments
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 65%] ␣
↪               190 alignments
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 70%] ␣
↪               190 alignments
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 75%] ␣
↪               190 alignments
2021-03-01 15:58:43,604 - phyluce_align_get_align_summary_data - INFO - [Matrix 80%] ␣
↪              29 alignments
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Matrix 85%] ␣
↪              29 alignments
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Matrix 90%] ␣
↪              29 alignments
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Matrix 95%] ␣
↪              29 alignments
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - --------------
↪---------- Character counts -----------------------
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'-' is present 7,819 times
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'?' is present 25,084 times
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'A' is present 70,371 times
2021-03-01 15:58:43,605 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'C' is present 48,431 times
2021-03-01 15:58:43,606 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'G' is present 41,941 times
2021-03-01 15:58:43,606 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'T' is present 74,642 times
2021-03-01 15:58:43,606 - phyluce_align_get_align_summary_data - INFO - =========␣
↪Completed phyluce_align_get_align_summary_data ========
```

Attention: Note that there are only 2 sets of counts in the `Data matrix completeness` section because (1) we dropped all loci having fewer than 3 taxa and (2) that only leaves two remaining options.

The most important data here are the number of loci we have and the number of loci in data matrices of different completeness. The locus length stats are also reasonably important, but they can also be misleading because edge-trimming does not remove internal gaps that often inflate the length of alignments.

Internal trimming

Now, let's do the same thing, but run internal trimming on the resulting alignments. We will do that by turning off trimming *–no-trim* and outputting FASTA formatted alignments with *–output-format fasta*.

```
# make sure we are in the correct directory
cd uce-tutorial/taxon-sets/all
```

(continues on next page)

(continued from previous page)

```
# align the data - turn off trimming and output FASTA
phyluce_align_seqcap_align \
    --input all-taxa-incomplete.fasta \
    --output mafft-nexus-internal-trimmed \
    --taxa 4 \
    --aligner mafft \
    --cores 12 \
    --incomplete-matrix \
    --output-format fasta \
    --no-trim \
    --log-path log
```

Attention: The number of UCE loci dropped here is abnormally large becase our **sample size is so small (n=4)**.

Warning: Note that I am using 12 physical CPU cores here. You need to use the number of physical cores available on *your* machine.

The output from the program should be the roughly the same as what we saw before. The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+-- assembly.conf
...
+-- taxon-sets
│       +-- all
│           +-- all-taxa-incomplete.conf
│           +-- all-taxa-incomplete.fasta
│           +-- all-taxa-incomplete.incomplete
│           +-- exploded-fastas
│           +-- log
│           +-- mafft-nexus-edge-trimmed
│           +-- mafft-nexus-internal-trimmed
│               +-- uce-1008.nexus
│               +-- uce-1014.nexus
│               +-- uce-1039.nexus
│               ...
│               +-- uce-991.nexus
...
+-- uce-search-results
```

Now, we are going to trim these loci using Gblocks:

```
# run gblocks trimming on the alignments
phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed \
    --alignments mafft-nexus-internal-trimmed \
    --output mafft-nexus-internal-trimmed-gblocks \
    --cores 12 \
    --log log
```

The output should look like this:

```
2021-03-01 16:01:38,320 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO -  Starting phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed
2021-03-01 16:01:38,321 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Version: 1.7.0
2021-03-01 16:01:38,321 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Commit: None
2021-03-01 16:01:38,321 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --alignments: /data/taxon-sets/all/mafft-nexus-internal-trimmed
2021-03-01 16:01:38,321 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --b1: 0.5
2021-03-01 16:01:38,321 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --b2: 0.85
2021-03-01 16:01:38,322 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --b3: 8
2021-03-01 16:01:38,322 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --b4: 10
2021-03-01 16:01:38,322 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --cores: 12
2021-03-01 16:01:38,322 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --input_format: fasta
2021-03-01 16:01:38,323 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --log_path: /data/taxon-sets/all/log
2021-03-01 16:01:38,323 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --output: /data/taxon-sets/all/mafft-nexus-internal-trimmed-
↪gblocks
2021-03-01 16:01:38,323 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --output_format: nexus
2021-03-01 16:01:38,323 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Argument --verbosity: INFO
2021-03-01 16:01:38,323 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Getting aligned sequences for trimming
2021-03-01 16:01:38,338 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Alignment trimming begins.
.................[continued]
2021-03-01 16:01:38,729 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Alignment trimming ends
2021-03-01 16:01:38,730 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO - Writing output files
2021-03-01 16:01:38,901 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- WARNING - Unable to write uce-816 - alignment too short
2021-03-01 16:01:39,099 - phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed␣
↪- INFO -  Completed phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed
```

The . values that you see represent loci that were aligned and succesfully trimmed. Any *X* values that you see represent loci that were aligned and trimmed so much that there was nothing left.

The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+── assembly.conf
...
+── taxon-sets
│       +── all
│           +── all-taxa-incomplete.conf
│           +── all-taxa-incomplete.fasta
│           +── all-taxa-incomplete.incomplete
│           +── exploded-fastas
│           +── log
```

(continues on next page)

(continued from previous page)

```
|           +-- mafft-nexus-edge-trimmed
|           +-- mafft-nexus-internal-trimmed
|           +-- mafft-nexus-internal-trimmed-gblocks
|               +-- uce-1008.nexus
|               +-- uce-1014.nexus
|               +-- uce-1039.nexus
|               ...
|               +-- uce-991.nexus
...
+-- uce-search-results
```

We can output summary stats for these alignments by running the following program:

```
phyluce_align_get_align_summary_data \
    --alignments mafft-nexus-internal-trimmed-gblocks \
    --cores 12 \
    --log-path log
```

Warning: Note that I am using 12 physical CPU cores here. You need to use the number of physical cores available on *your* machine.

The output from the program should look like:

```
2021-03-01 16:03:34,322 - phyluce_align_get_align_summary_data - INFO - =========␣
↪Starting phyluce_align_get_align_summary_data =========
2021-03-01 16:03:34,322 - phyluce_align_get_align_summary_data - INFO - Version: 1.7.0
2021-03-01 16:03:34,322 - phyluce_align_get_align_summary_data - INFO - Commit: None
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪alignments: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪cores: 12
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪input_format: nexus
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪log_path: /data/taxon-sets/all/log
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪output: None
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪show_taxon_counts: False
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Argument --
↪verbosity: INFO
2021-03-01 16:03:34,323 - phyluce_align_get_align_summary_data - INFO - Getting␣
↪alignment files
2021-03-01 16:03:34,334 - phyluce_align_get_align_summary_data - INFO - Computing␣
↪summary statistics using 12 cores
2021-03-01 16:03:34,641 - phyluce_align_get_align_summary_data - INFO - --------------
↪--------- Alignment summary -----------------------
2021-03-01 16:03:34,642 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪loci:  189
2021-03-01 16:03:34,642 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪length:        70,380
2021-03-01 16:03:34,643 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪mean:  372.38
2021-03-01 16:03:34,643 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪95% CI:        21.61
```

(continues on next page)

(continued from previous page)

```
2021-03-01 16:03:34,643 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪min:   140
2021-03-01 16:03:34,643 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪max:   797
2021-03-01 16:03:34,644 - phyluce_align_get_align_summary_data - INFO - --------------
↪----- Informative Sites summary -------------------
2021-03-01 16:03:34,645 - phyluce_align_get_align_summary_data - INFO - [Sites] loci:␣
↪       189
2021-03-01 16:03:34,645 - phyluce_align_get_align_summary_data - INFO - [Sites]␣
↪total:      69
2021-03-01 16:03:34,645 - phyluce_align_get_align_summary_data - INFO - [Sites] mean:␣
↪       0.37
2021-03-01 16:03:34,645 - phyluce_align_get_align_summary_data - INFO - [Sites] 95%␣
↪CI:      0.19
2021-03-01 16:03:34,646 - phyluce_align_get_align_summary_data - INFO - [Sites] min: ␣
↪       0
2021-03-01 16:03:34,646 - phyluce_align_get_align_summary_data - INFO - [Sites] max: ␣
↪       12
2021-03-01 16:03:34,648 - phyluce_align_get_align_summary_data - INFO - --------------
↪----------- Taxon summary -------------------------
2021-03-01 16:03:34,649 - phyluce_align_get_align_summary_data - INFO - [Taxa] mean: ␣
↪               3.15
2021-03-01 16:03:34,649 - phyluce_align_get_align_summary_data - INFO - [Taxa] 95%␣
↪CI:       0.05
2021-03-01 16:03:34,649 - phyluce_align_get_align_summary_data - INFO - [Taxa] min:  ␣
↪       3
2021-03-01 16:03:34,650 - phyluce_align_get_align_summary_data - INFO - [Taxa] max:  ␣
↪       4
2021-03-01 16:03:34,650 - phyluce_align_get_align_summary_data - INFO - --------------
↪--- Missing data from trim summary ----------------
2021-03-01 16:03:34,651 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪mean:      0.00
2021-03-01 16:03:34,651 - phyluce_align_get_align_summary_data - INFO - [Missing] 95%␣
↪CI:    0.00
2021-03-01 16:03:34,651 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪min:       0.00
2021-03-01 16:03:34,651 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪max:       0.00
2021-03-01 16:03:34,655 - phyluce_align_get_align_summary_data - INFO - --------------
↪------ Character count summary --------------------
2021-03-01 16:03:34,655 - phyluce_align_get_align_summary_data - INFO - [All␣
↪characters]     222,814
2021-03-01 16:03:34,655 - phyluce_align_get_align_summary_data - INFO - [Nucleotides]␣
↪               215,091
2021-03-01 16:03:34,656 - phyluce_align_get_align_summary_data - INFO - --------------
↪-- Data matrix completeness summary ---------------
2021-03-01 16:03:34,656 - phyluce_align_get_align_summary_data - INFO - [Matrix 50%] ␣
↪               189 alignments
2021-03-01 16:03:34,656 - phyluce_align_get_align_summary_data - INFO - [Matrix 55%] ␣
↪               189 alignments
2021-03-01 16:03:34,657 - phyluce_align_get_align_summary_data - INFO - [Matrix 60%] ␣
↪               189 alignments
2021-03-01 16:03:34,657 - phyluce_align_get_align_summary_data - INFO - [Matrix 65%] ␣
↪               189 alignments
2021-03-01 16:03:34,657 - phyluce_align_get_align_summary_data - INFO - [Matrix 70%] ␣
↪               189 alignments
2021-03-01 16:03:34,657 - phyluce_align_get_align_summary_data - INFO - [Matrix 75%] ␣
↪               189 alignments
```

(continues on next page)

(continued from previous page)

```
2021-03-01 16:03:34,658 - phyluce_align_get_align_summary_data - INFO - [Matrix 80%] ␣
↪               29 alignments
2021-03-01 16:03:34,658 - phyluce_align_get_align_summary_data - INFO - [Matrix 85%] ␣
↪               29 alignments
2021-03-01 16:03:34,658 - phyluce_align_get_align_summary_data - INFO - [Matrix 90%] ␣
↪               29 alignments
2021-03-01 16:03:34,658 - phyluce_align_get_align_summary_data - INFO - [Matrix 95%] ␣
↪               29 alignments
2021-03-01 16:03:34,659 - phyluce_align_get_align_summary_data - INFO - --------------
↪---------- Character counts -----------------------
2021-03-01 16:03:34,659 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'-' is present 7,723 times
2021-03-01 16:03:34,659 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'A' is present 64,167 times
2021-03-01 16:03:34,659 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'C' is present 44,488 times
2021-03-01 16:03:34,660 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'G' is present 37,901 times
2021-03-01 16:03:34,660 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'T' is present 68,535 times
2021-03-01 16:03:34,660 - phyluce_align_get_align_summary_data - INFO - =========␣
↪Completed phyluce_align_get_align_summary_data ========
```

Alignment cleaning

If you look in one of the files for the alignments we currently have, you will notice that each alignment contains a name that is a combination of the taxon name + the locus name for that taxon. This is not what we want downstream, but it does enable us to ensure the correct data went into each alignment. So, we need to clean our alignments. For the remainder of this tutorial, we will work with the Gblocks trimmed alignments, so we will clean those alignments:

```
# make sure we are in the correct directory
cd uce-tutorial/taxon-sets/all

# align the data - turn off trimming and output FASTA
phyluce_align_remove_locus_name_from_files \
    --alignments mafft-nexus-internal-trimmed-gblocks \
    --output mafft-nexus-internal-trimmed-gblocks-clean \
    --cores 12 \
    --log-path log
```

The output should be similar to:

```
2021-03-01 16:05:12,605 - phyluce_align_remove_locus_name_from_files - INFO - ===␣
↪Starting phyluce_align_remove_locus_name_from_files ===
2021-03-01 16:05:12,605 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Version: 1.7.0
2021-03-01 16:05:12,605 - phyluce_align_remove_locus_name_from_files - INFO - Commit:␣
↪None
2021-03-01 16:05:12,606 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --alignments: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks
2021-03-01 16:05:12,606 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --cores: 12
2021-03-01 16:05:12,606 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --input_format: nexus
2021-03-01 16:05:12,607 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --log_path: /data/taxon-sets/all/log
```

(continues on next page)

(continued from previous page)

```
2021-03-01 16:05:12,607 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --output: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks-clean
2021-03-01 16:05:12,607 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --output_format: nexus
2021-03-01 16:05:12,608 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --taxa: None
2021-03-01 16:05:12,608 - phyluce_align_remove_locus_name_from_files - INFO -␣
↪Argument --verbosity: INFO
2021-03-01 16:05:12,608 - phyluce_align_remove_locus_name_from_files - INFO - Getting␣
↪alignment files
Running...............[continued]
2021-03-01 16:05:12,848 - phyluce_align_remove_locus_name_from_files - INFO - Taxon␣
↪names in alignments: gallus_gallus,mus_musculus,anolis_carolinensis,alligator_
↪mississippiensis
2021-03-01 16:05:12,849 - phyluce_align_remove_locus_name_from_files - INFO - ===␣
↪Completed phyluce_align_remove_locus_name_from_files ==
```

The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+── assembly.conf
...
+── taxon-sets
│       +── all
│           +── all-taxa-incomplete.conf
│           +── all-taxa-incomplete.fasta
│           +── all-taxa-incomplete.incomplete
│           +── exploded-fastas
│           +── log
│           +── mafft-nexus-edge-trimmed
│           +── mafft-nexus-internal-trimmed
│           +── mafft-nexus-internal-trimmed-gblocks
│           +── mafft-nexus-internal-trimmed-gblocks-clean
│               +── uce-1008.nexus
│               +── uce-1014.nexus
│               +── uce-1039.nexus
│               ...
│               +── uce-991.nexus
...
+── uce-search-results
```

Now, if you look in one of the individual alignment files, you will see that the locus names are removed. We're ready to generate our final data matrices.

Final data matrices

For the most part, I analyze 75% and 95% complete matrices, where "completeness" for the 75% matrix means that, in a study of 100 taxa (total), all alignments will contain at least 75 of these 100 taxa. Similarly, for the 95% matrix, in a study of 100 taxa, all alignments will contain 95 of these 100 taxa.

Attention: Notice that this metric for completeness does not pay attention to which taxa are in which alignments - so the 75%, above, does **not** mean that a given taxon will have data in all 75 of 100 alignments.

To create a 75% data matrix, run the following. Notice that the integer following *–taxa* is the **total** number of organisms in the study.

```
# make sure we are in the correct directory
cd uce-tutorial/taxon-sets/all

# the integer following --taxa is the number of TOTAL taxa
# and I use "75p" to denote the 75% complete matrix
phyluce_align_get_only_loci_with_min_taxa \
    --alignments mafft-nexus-internal-trimmed-gblocks-clean \
    --taxa 4 \
    --percent 0.75 \
    --output mafft-nexus-internal-trimmed-gblocks-clean-75p \
    --cores 12 \
    --log-path log
```

The output should look like the following:

```
2021-03-01 16:06:56,294 - phyluce_align_get_only_loci_with_min_taxa - INFO - =======␣
↪Starting phyluce_align_get_only_loci_with_min_taxa ======
2021-03-01 16:06:56,294 - phyluce_align_get_only_loci_with_min_taxa - INFO - Version:␣
↪1.7.0
2021-03-01 16:06:56,294 - phyluce_align_get_only_loci_with_min_taxa - INFO - Commit:␣
↪None
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--alignments: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks-clean
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--cores: 12
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--input_format: nexus
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--log_path: /data/taxon-sets/all/log
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--output: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks-clean-75p
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--percent: 0.75
2021-03-01 16:06:56,295 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--taxa: 4
2021-03-01 16:06:56,296 - phyluce_align_get_only_loci_with_min_taxa - INFO - Argument␣
↪--verbosity: INFO
2021-03-01 16:06:56,296 - phyluce_align_get_only_loci_with_min_taxa - INFO - Getting␣
↪alignment files
2021-03-01 16:06:56,534 - phyluce_align_get_only_loci_with_min_taxa - INFO - Copied␣
↪189 alignments of 189 total containing ≥ 0.75 proportion of taxa (n = 3)
2021-03-01 16:06:56,534 - phyluce_align_get_only_loci_with_min_taxa - INFO - ======␣
↪Completed phyluce_align_get_only_loci_with_min_taxa ======
```

The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+-- assembly.conf
...
+-- taxon-sets
|       +-- all
|           +-- all-taxa-incomplete.conf
|           +-- all-taxa-incomplete.fasta
|           +-- all-taxa-incomplete.incomplete
|           +-- exploded-fastas
|           +-- log
```

(continues on next page)

(continued from previous page)

```
│               +── mafft-nexus-edge-trimmed
│               +── mafft-nexus-internal-trimmed
│               +── mafft-nexus-internal-trimmed-gblocks
│               +── mafft-nexus-internal-trimmed-gblocks-clean
│               +── mafft-nexus-internal-trimmed-gblocks-clean-75p
│                   +── uce-1008.nexus
│                   +── uce-1014.nexus
│                   +── uce-1039.nexus
│                   ...
│                   +── uce-991.nexus
...
+── uce-search-results
```

Preparing data for downstream analysis

Now that we have our *75p* data matrix completed, we can generate input files for subsequent phylogenetic analysis. For the most part, I use RAxML or IQTree, both of which will take a phylip-formatted file as input. Formatting our *75p* data into a phylip file for these programs is rather easy. To do that, run:

```
# make sure we are in the correct directory
cd uce-tutorial/taxon-sets/all

# build the concatenated data matrix
phyluce_align_concatenate_alignments \
    --alignments mafft-nexus-internal-trimmed-gblocks-clean-75p \
    --output mafft-nexus-internal-trimmed-gblocks-clean-75p-raxml \
    --phylip \
    --log-path log
```

The output from this program will look like:

```
2021-03-01 16:09:01,391 - phyluce_align_concatenate_alignments - INFO - =========␣
↪Starting phyluce_align_concatenate_alignments =========
2021-03-01 16:09:01,391 - phyluce_align_concatenate_alignments - INFO - Version: 1.7.0
2021-03-01 16:09:01,391 - phyluce_align_concatenate_alignments - INFO - Commit: None
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪alignments: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks-clean-75p
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪input_format: nexus
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪log_path: /data/taxon-sets/all/log
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪nexus: False
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪output: /data/taxon-sets/all/mafft-nexus-internal-trimmed-gblocks-clean-75p-raxml
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪phylip: True
2021-03-01 16:09:01,392 - phyluce_align_concatenate_alignments - INFO - Argument --
↪verbosity: INFO
2021-03-01 16:09:01,393 - phyluce_align_concatenate_alignments - INFO - Reading input␣
↪alignments in NEXUS format
2021-03-01 16:09:01,393 - phyluce_align_concatenate_alignments - INFO - Getting␣
↪alignment files
2021-03-01 16:09:01,733 - phyluce_align_concatenate_alignments - INFO - Concatenating␣
↪files
```

(continues on next page)

(continued from previous page)

```
2021-03-01 16:09:01,796 - phyluce_align_concatenate_alignments - INFO - Writing␣
↪concatenated alignment to PHYLIP format (with charsets)
2021-03-01 16:09:01,803 - phyluce_align_concatenate_alignments - INFO - =========␣
↪Completed phyluce_align_concatenate_alignments ========
```

Attention: Charsets are now output by default for all data sets. You generally want these and the cost for them is low.

The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+── assembly.conf
...
+── taxon-sets
│       +── all
│           +── all-taxa-incomplete.conf
│           +── all-taxa-incomplete.fasta
│           +── all-taxa-incomplete.incomplete
│           +── exploded-fastas
│           +── log
│           +── mafft-nexus-edge-trimmed
│           +── mafft-nexus-internal-trimmed
│           +── mafft-nexus-internal-trimmed-gblocks
│           +── mafft-nexus-internal-trimmed-gblocks-clean
│           +── mafft-nexus-internal-trimmed-gblocks-clean-75p
│           +── mafft-nexus-internal-trimmed-gblocks-clean-75p-raxml
│               +── mafft-nexus-internal-trimmed-gblocks-clean-75p.charsets
│               +── mafft-nexus-internal-trimmed-gblocks-clean-75p.phylip
...
+── uce-search-results
```

If you need to output concatenated data in `NEXUS` format, you can simply run the same program as above, but change `--phylip` to `--nexus`, like:

```
# make sure we are in the correct directory
cd uce-tutorial/taxon-sets/all

# build the concatenated data matrix
phyluce_align_concatenate_alignments \
    --alignments mafft-nexus-internal-trimmed-gblocks-clean-75p \
    --output mafft-nexus-internal-trimmed-gblocks-clean-75p-raxml \
    --nexus \
    --log-path log
```

Downstream Analysis

The above data are ready to analyze in a program like RAxML or IQTree. See the documentation for those programs to learn how to use them.

If you are performing locus-based inference (e.g. so-called gene-tree, species-tree methods, you can analyze the individual locus alignments present in the folder you created just prior to concatenation (e.g. `mafft-nexus-internal-trimmed-gblocks-clean-75p`). If you need to convert those files into another format (by default, they are in `NEXUS` format), you can use phyluce_align_convert_one_align_to_another. For performing analyses to infer a locus-tree from individual alignments, you might look into a program like pargenes. You can also reasonably script IQTree to do this on, for example, an HPC system (we do it using GNU Parallel to send each alignment to one compute core, where we run IQTree on that alignment).

Next Steps

After completing the tutorial, you should have a reasonably good idea of how to use phyluce in a day-to-day situation. If you want to know more about specifics, you can read through the *Phyluce in Daily Use* sections, which provide additional detail. Also be sure to poke around the other programs that come with phyluce - short list of which you can find in the *List of Phyluce Programs*.

3.3.2 Tutorial II: Phasing UCE data

The following workflow derives from Andermann et al. 2018 (https://doi.org/10.1093/sysbio/syy039) and focuses on phasing SNPs in UCE data.

To phase your UCE data, you need to have individual-specific "reference" contigs against which to align your raw reads. Generally speaking, you can create these individual-specific reference contigs at several stages of the phyluce pipeline, and the stage at which you choose to do this may depend on the analyses that you are running. That said, I think that the best way to proceed uses edge-trimmed exploded alignments as your reference contigs, aligns raw reads to those, and uses the exploded alignments and raw reads to phase your data.

Attention: We have not implemented code that you can use if you are trimming your alignment data with some other approach (e.g. gblocks or trimal).

Exploding aligned and trimmed UCE sequences

Probably the best way to proceed (you can come up with other ways to do this) is to choose loci that have already been aligned and edge-trimmed as the basis for SNP calling and haplotype phasing. The benefit of this approach is that the individual-specific reference contigs you are inputting to the process will be somewhat normalized across all of your individuals because you have already generated alignments from all of your UCE loci and trimmed the edges of these loci.

To follow this approach, first proceed through the *Edge trimming* section of *Tutorial I: UCE Phylogenomics*. Then, you can "explode" the directory of alignments you have generated to create separate FASTA files for each individual using the following (this assumes your alignments are in *mafft-nexus-edge-trimmed* as in the tutorial).

```
# explode the alignment files in mafft-nexus-edge-trimmed by taxon create a taxon-
↪specific FASTA
phyluce_align_explode_alignments \
    --alignments mafft-nexus-edge-trimmed \
    --input-format nexus \
    --output mafft-nexus-edge-trimmed-exploded \
    --by-taxon
```

The current directory structure should look like (I've collapsed a number of branches in the tree):

```
uce-tutorial
+── assembly.conf
...
+── taxon-sets
```

(continues on next page)

(continued from previous page)

```
|       +-- all
|           +-- all-taxa-incomplete.conf
|           +-- all-taxa-incomplete.fasta
|           +-- all-taxa-incomplete.incomplete
|           +-- exploded-fastas
|           +-- log
|           +-- mafft-nexus-edge-trimmed
|           +-- mafft-nexus-edge-trimmed-exploded
|               +-- alligator_mississippiensis.fasta
|               +-- anolis_carolinensis.fasta
|               +-- gallus_gallus.fasta
|               +-- mus_musculus.fasta
...
+-- uce-search-results
```

You may want to get stats on these exploded-fastas by running something like the following:

```
# get summary stats on the FASTAS
for i in mafft-nexus-edge-trimmed-exploded/*.fasta;
do
    phyluce_assembly_get_fasta_lengths --input $i --csv;
done
```

Creating a re-alignment configuration file

Before aligning raw reads back to these reference contigs using `bwa`, you have to create a configuration file, which tells the program where the cleaned and trimmed fastq reads are stored for each sample and where to find the reference FASTA file for each sample. The configuration file should look like in the following example and should be saved as e.g. `phasing.conf`

```
[references]
alligator_mississippiensis:/Users/bcf/tmp/phyluce/mafft-nexus-edge-trimmed-exploded/
↪alligator_mississippiensis.fasta
anolis_carolinensis:/Users/bcf/tmp/phyluce/mafft-nexus-edge-trimmed-exploded/anolis_
↪carolinensis.fasta
gallus_gallus:/Users/bcf/tmp/phyluce/mafft-nexus-edge-trimmed-exploded/gallus_gallus.
↪fasta
mus_musculus:/Users/bcf/tmp/phyluce/mafft-nexus-edge-trimmed-exploded/mus_musculus.
↪fasta

[individuals]
alligator_mississippiensis:/Users/bcf/tmp/phyluce/clean-fastq/alligator_
↪mississippiensis/split-adapter-quality-trimmed/
anolis_carolinensis:/Users/bcf/tmp/phyluce/clean-fastq/anolis_carolinensis/split-
↪adapter-quality-trimmed/
gallus_gallus:/Users/bcf/tmp/phyluce/clean-fastq/gallus_gallus/split-adapter-quality-
↪trimmed/
mus_musculus:/Users/bcf/tmp/phyluce/clean-fastq/mus_musculus/split-adapter-quality-
↪trimmed/

[flowcell]
alligator_mississippiensis:D1HTMACXX
anolis_carolinensis:C0DBPACXX
gallus_gallus:A8E3E
mus_musculus:C0DBPACXX
```

[references]

In this section you simply state the sample ID (`genus_species1`) followed by a colon (`:`) and the full path to the sample-specific FASTA library which was generated in the previous step.

[individuals]

In this section you give the complete path to the cleaned and trimmed reads folder for each sample.

> **Attention:** The cleaned reads used by this program should be generated by illumiprocessor because the folder structure of the cleaned reads files is assumed to be that of illumiprocessor . This means that the zipped fastq files (fastq.gz) have to be located in a subfolder with the name `split-adapter-quality-trimmed` within each sample-specific folder.

[flowcell]

The flowcell section is meant to add flowcell information from the Illumina run to the header to the BAM file that is created. This can be helpful for later identication of sample and run information. If you do not know the flowcell information for the data you are processing, you can look inside of the *fastq.gz* file using a program like *less*. The flowcell identifier is the set of digits and numbers after the 2nd colon.

```
@J00138:149:HT23LBBXX:8:1101:5589:1015 1:N:0:ATAAGGCG+CATACCAC
            ^^^^^^^^^
```

Alternatively, you can enter any string of information here (no spaces) that you would like to help identify a given sample (e.g. `XXYYZZ`).

Mapping reads against contigs

To map the fastq read files against the contig reference database for each sample, run the folliwing. This will use `bwa mem` to map the raw reads to the "reference" contigs:

```
phyluce_snp_bwa_multiple_align \
    --config phasing.conf \
    --output multialign-bams \
    --cores 12 \
    --log-path log \
    --mem
```

This will produce an output along these lines

```
2016-03-09 16:40:22,628 - phyluce_snp_bwa_multiple_align - INFO - ============␣
↪Starting phyluce_snp_bwa_multiple_align ============
2016-03-09 16:40:22,628 - phyluce_snp_bwa_multiple_align - INFO - Version: 1.5.0
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --config: /
↪path/to/phasing.conf
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --cores: 1
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --log_
↪path: None
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --mem:␣
↪False
```

(continues on next page)

(continued from previous page)

```
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --no_
↪remove_duplicates: False
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --output: /
↪path/to/mapping_results
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --
↪subfolder: split-adapter-quality-trimmed
2016-03-09 16:40:22,629 - phyluce_snp_bwa_multiple_align - INFO - Argument --
↪verbosity: INFO
2016-03-09 16:40:22,630 - phyluce_snp_bwa_multiple_align - INFO - ============␣
↪Starting phyluce_snp_bwa_multiple_align ============
2016-03-09 16:40:22,631 - phyluce_snp_bwa_multiple_align - INFO - Getting input␣
↪filenames and creating output directories
2016-03-09 16:40:22,633 - phyluce_snp_bwa_multiple_align - INFO - --------------------
↪-- Processing genus_species1 ----------------------
2016-03-09 16:40:22,633 - phyluce_snp_bwa_multiple_align - INFO - Finding fastq/fasta␣
↪files
2016-03-09 16:40:22,636 - phyluce_snp_bwa_multiple_align - INFO - File type is fastq
2016-03-09 16:40:22,637 - phyluce_snp_bwa_multiple_align - INFO - Creating read index␣
↪file for genus_species1-READ1.fastq.gz
2016-03-09 16:40:33,999 - phyluce_snp_bwa_multiple_align - INFO - Creating read index␣
↪file for genus_species1-READ2.fastq.gz
2016-03-09 16:40:45,142 - phyluce_snp_bwa_multiple_align - INFO - Building BAM for␣
↪genus_species1
2016-03-09 16:41:33,195 - phyluce_snp_bwa_multiple_align - INFO - Cleaning BAM for␣
↪genus_species1
2016-03-09 16:42:03,410 - phyluce_snp_bwa_multiple_align - INFO - Adding RG header to␣
↪BAM for genus_species1
2016-03-09 16:42:49,518 - phyluce_snp_bwa_multiple_align - INFO - Marking read␣
↪duplicates from BAM for genus_species1
2016-03-09 16:43:26,917 - phyluce_snp_bwa_multiple_align - INFO - Creating read index␣
↪file for genus_species1-READ-singleton.fastq.gz
2016-03-09 16:43:27,066 - phyluce_snp_bwa_multiple_align - INFO - Building BAM for␣
↪genus_species1
2016-03-09 16:43:27,293 - phyluce_snp_bwa_multiple_align - INFO - Cleaning BAM for␣
↪genus_species1
2016-03-09 16:43:27,748 - phyluce_snp_bwa_multiple_align - INFO - Adding RG header to␣
↪BAM for genus_species1
2016-03-09 16:43:28,390 - phyluce_snp_bwa_multiple_align - INFO - Marking read␣
↪duplicates from BAM for genus_species1
2016-03-09 16:43:30,633 - phyluce_snp_bwa_multiple_align - INFO - Merging BAMs for␣
↪genus_species1
2016-03-09 16:44:05,811 - phyluce_snp_bwa_multiple_align - INFO - Indexing BAM for␣
↪genus_species1
2016-03-09 16:44:08,047 - phyluce_snp_bwa_multiple_align - INFO - --------------------
↪-- Processing genus_species2 ----------------------
...
```

Phasing mapped reads

In the previous step you aligned your sequence reads against the reference FASTA file for each sample. The results are stored in the output folder in `bam` format. Now you can start the actual phasing of the reads. This will analyze and sort the reads within each bam file into two separate bam files (`genus_species1.0.bam` and `genus_species1.1.bam`).

The program is very easy to run and just requires the path to the bam files (output folder from previous mapping program, `/path/to/mapping_results`) and the path to the configuration file, which is the same file as used in the previous step (/path/to/phasing.conf). Then, run:

```
phyluce_snp_phase_uces \
    --config phasing.conf \
    --bams multialign-bams \
    --output multialign-bams-phased-reads
```

The output will look something like the following

```
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - ================ Starting␣
↪phyluce_snp_phase_uces ================
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - Version: git 0babc1a
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - Argument --bams: /Users/bcf/
↪tmp/phyluce/multialign-bams
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - Argument --config: /Users/
↪bcf/tmp/phyluce/phasing.conf
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - Argument --conservative:␣
↪False
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - Argument --cores: 1
2018-07-27 09:58:35,963 - phyluce_snp_phase_uces - INFO - Argument --log_path: None
2018-07-27 09:58:35,964 - phyluce_snp_phase_uces - INFO - Argument --output: /Users/
↪bcf/tmp/phyluce/multialign-bams-phased-reads
2018-07-27 09:58:35,964 - phyluce_snp_phase_uces - INFO - Argument --verbosity: INFO
2018-07-27 09:58:35,964 - phyluce_snp_phase_uces - INFO - ================ Starting␣
↪phyluce_snp_phase_uces ================
2018-07-27 09:58:35,964 - phyluce_snp_phase_uces - INFO - Getting input filenames and␣
↪creating output directories
2018-07-27 10:02:10,526 - phyluce_snp_phase_uces - INFO - ================ Starting␣
↪phyluce_snp_phase_uces ================
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Version: git 0babc1a
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --bams: /Users/bcf/
↪tmp/phyluce/multialign-bams
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --config: /Users/
↪bcf/tmp/phyluce/phasing.conf
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --conservative:␣
↪False
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --cores: 1
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --log_path: None
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --output: /Users/
↪bcf/tmp/phyluce/multialign-bams-phased-reads
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - Argument --verbosity: INFO
2018-07-27 10:02:10,527 - phyluce_snp_phase_uces - INFO - ================ Starting␣
↪phyluce_snp_phase_uces ================
2018-07-27 10:02:10,528 - phyluce_snp_phase_uces - INFO - Getting input filenames and␣
↪creating output directories
2018-07-27 10:02:10,528 - phyluce_snp_phase_uces - INFO - ------------- Processing␣
↪alligator_mississippiensis -------------
2018-07-27 10:02:10,528 - phyluce_snp_phase_uces - INFO - Phasing BAM file for␣
↪alligator_mississippiensis
2018-07-27 10:02:21,695 - phyluce_snp_phase_uces - INFO - Sorting BAM for alligator_
↪mississippiensis
2018-07-27 10:02:23,115 - phyluce_snp_phase_uces - INFO - Sorting BAM for alligator_
↪mississippiensis
2018-07-27 10:02:24,533 - phyluce_snp_phase_uces - INFO - Creating REF/ALT allele␣
↪FASTQ file 0
2018-07-27 10:02:24,583 - phyluce_snp_phase_uces - INFO - Creating REF/ALT allele␣
↪FASTQ file 1
2018-07-27 10:02:24,613 - phyluce_snp_phase_uces - INFO - Creating REF/ALT allele␣
↪FASTQ file unphased
```

(continues on next page)

(continued from previous page)

```
2018-07-27 10:02:24,643 - phyluce_snp_phase_uces - INFO - Creating REF/ALT allele␣
↪FASTA file 0 from FASTQ 0
2018-07-27 10:02:24,654 - phyluce_snp_phase_uces - INFO - Creating REF/ALT allele␣
↪FASTA file 1 from FASTQ 1
2018-07-27 10:02:24,662 - phyluce_snp_phase_uces - INFO - Creating REF/ALT allele␣
↪FASTA file unphased from FASTQ unphased
2018-07-27 10:02:24,673 - phyluce_snp_phase_uces - INFO - Checking for correct FASTA␣
↪files
2018-07-27 10:02:24,673 - phyluce_snp_phase_uces - INFO - Cleaning FASTA files
2018-07-27 10:02:24,681 - phyluce_snp_phase_uces - INFO - Balancing FASTA files
2018-07-27 10:02:24,682 - phyluce_snp_phase_uces - INFO - Symlinking FASTA files
```

The program automatically produces a consensus sequence for each of these phased bam files (= allele sequence) and stores these allele sequences of all samples in a joined FASTA file (`joined_allele_sequences_all_samples.fasta`). This allele FASTA is deposited in the subfolder `fastas` within your output folder (e.g. `/path/to/multialign-bams-phased-reads/fastas`).

You can directly input that file (`joined_allele_sequences_all_samples.fasta`) back into the alignment pipeline, like so:

```
phyluce_align_seqcap_align \
    --fasta /path/to/multialign-bams-phased-reads/fastas/joined_allele_sequences_all_
↪samples.fasta \
    --output PHASED-DATA mafft-nexus-edge-trimmed \
    --taxa 4 \
    --aligner mafft \
    --cores 12 \
    --incomplete-matrix \
    --log-path log
```

Following alignment, you can choose how you'd like to treat these data (e.g. internally trim, analyze, etc).

3.3.3 Tutorial III: Harvesting UCE Loci From Genomes

In many cases, genomic data exist for some (or many) taxa, and you want to "harvest" those loci from the genome(s) available to you for inclusion in a study. This tutorial is meant to explain how to do this. For this example, we'll download two genome sequences from web repositories, locate, and extract UCE loci from these genomes for subsequent analysis.

The taxa we are working with will be:

- Gallus gallus
- Alligator mississippiensis

Starting directory structure

To keep things clear, we're going to assume you are working in some directory, which I'll call *uce-genome*. We'll be working from the top of this directory in the steps below:

```
uce-genome
```

Download the data

You can download genome assemblies from any number of sources - some better than others. Here, we're going to download one genome assembly (chicken; galGal4) from the UCSC Genome Browser and another (alligator) from NCBI. We're using two difference sources so you can see some of the differences in the process. . . and what you might need to do in order to "clean up" a given genome sequence. We'll start with the easy one.

chicken (galGal4)

The UCSC Genome Browser is a great resource for lots of data that is easy to find and easy to use. In particular, we're interested in the UCSC Genome Browser Downloads area, where you can find genome sequence, genome annotations, etc. for many model (and non-model) taxa. In this case, we want the *galGal4* genome sequence, which is the 4th "official" assembly of the chicken genome sequence (AKA *GCA_000002315.2*). You can find this by navigating from the UCSC Genome Browser Downloads to the Chicken section of the page. Under the *galGal4* heading, click on Full data set. This will take you to the data download page for *galGal4* where there is a listing of all the data you can download for the *galGal4* assembly. We're interested in the *galGal4.2bit* file, which is a compressed representation of the genome in 2bit format. You can either click on this file name to download the file, or navigate to your *uce- genome* folder and:

```
$ cd uce-genome
$ mkdir galGal4
$ cd galGal4
wget http://hgdownload.soe.ucsc.edu/goldenPath/galGal4/bigZips/galGal4.2bit
```

This put a 2bit file in our *uce-genome* directory, so that our directory structure looks like:

```
uce-genome
+-- galGal4
    +-- galGal4.2bit
```

There are various utilities for dealing with 2bit files that you can download as part of the Kent Source Archive, a set of programs for dealing with genome-scale data. Probably the most important of these are *faToTwoBit*, which we use below; *twoBitInfo*, which gives us information on a given *2bit* file; and *twoBitToFa* which converts a *2bit* file back to FASTA format. If we run *twoBitInfo* against *galGal4.2bit*, we see something like:

```
$ cd uce-genome/galGal4
$ twoBitInfo galGal4.2bit sizes.tab

$ head -n 5 sizes.tab
chr1    195276750
chr10   19911089
chr10_AADN03010416_random   11080
chr10_AADN03010420_random   8415
chr10_JH375184_random   3009
```

which shows us the scaffolds/contigs and their sizes.

alligator (allMis1)

Although you can get the alligator genomes from UCSC, we'll download it from NCBI, so that you can see the differences in the process/data. Explaining NCBI is beyond the scope of what we're doing here, so we'll just navigate to the NCBI alligator genome assembly page. If you click the Assembly link on the right hand side of the page (under "Related information"), this will take you to the suite of assemblies for alligator.

We'll download the *Algmis_Hirise_1.0* assembly, which is an improvement on the original assembly. To do this, click on the *Algmis_Hirise_1.0* link, and look for the Download the GenBank assembly link on the top right corner of the next page. This will take you to an FTP page, where you want to download GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna.gz. You can do this by clicking on the link or by:

```
$ cd uce-genome
$ mkdir allMis2
$ cd allMis2
```

wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/001/541/155/GCA_001541155.1_Algmis_Hirise_1.0/GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna.gz

This put a gzipped fasta file in our *uce-genome/allMis2* directory, so that our directory structure looks like:

```
uce-genome
+-- allMis2
│   +-- GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna.gz
+-- galGal4
    +-- galGal4.2bit
    +-- sizes.tab
```

Now, we need to unzip this and have a look at the file:

```
$ cd uce-genome/allMis2
$ gunzip GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna.gz

# take a look at the contents of this file:
$ head -n 1 GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna
>LPUV01000001.1 Alligator mississippiensis ScZkoYb_3522, whole genome shotgun sequence
```

Note that the header has a lot of text in it, and this text is not always good. We're going to convert this file to *2bit* format using *faToTwoBit* from the Kent Source Archive, which will remove everything following the first space in the header line:

```
$ faToTwoBit GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna allMis2.2bit
$ twoBitInfo allMis2.2bit sizes.tab
$ head -n 5 sizes.tab
LPUV01000001.1  1279
LPUV01000002.1  4883
LPUV01000003.1  1105
LPUV01000004.1  22955
LPUV01000005.1  1137523
```

Now that we've converted the fasta file to *2bit* format, we can delete the fasta file:

```
$ rm GCA_001541155.1_Algmis_Hirise_1.0_genomic.fna
```

And your directory structure should look like:

```
uce-genome
+-- allMis2
│   +-- allMis2.2bit
│   +-- sizes.tab
+-- galGal4
    +-- galGal4.2bit
    +-- sizes.tab
```

Attention: It's easiest to use *2bit* files of each genome you want to search for UCE loci.

Finding UCE loci

Now that we've downloaded our genome assemblies, it's time to find those contigs/scaffolds in the assembly that contain UCE loci.

Get the UCE probes

To do that, we need to get the UCE probe set we want to search for into our directory. Here, we'll search for the UCE 5k probe set. However, you can use whichever probe set you like - you just need to know the path to this file.

```
> cd uce-genome
# download the probe set
wget https://raw.githubusercontent.com/faircloth-lab/uce-probe-sets/master/uce-5k-
↪probe-set/uce-5k-probes.fasta
```

Now, our directory structure looks like:

```
uce-genome
+-- allMis2
|    +-- allMis2.2bit
|    +-- sizes.tab
+-- galGal4
|    +-- galGal4.2bit
|    +-- sizes.tab
+-- uce-5k-probes.fasta
```

Align the probes to the genomes

We need to align the probes to the genome sequences. The program that we are going to run assumes that each *2bit* genome file is in it's own directory (which we have ensured, above).

Attention: If you use some other organizational structure, you still need to ensure that each genome sequence is in it's own directory. So, if you have some directory *genomes*, the genomes must be organized within that folder like:

```
genomes
+-- allMis2
|    +-- allMis2.2bit
+-- galGal4
     +-- galGal4.2bit
```

Now, we need to think of some name for the database to create (here *tutorial3.sqlite*), the name of an output in which to store the *lastz* search results, the path to the genome sequences, the name of the genome sequences, the path to the probe file, and a number of compute cores to use:

```
> cd uce-genome

# run the search
```

(continues on next page)

(continued from previous page)

```
> phyluce_probe_run_multiple_lastzs_sqlite \
    --db tutorial3.sqlite \
    --output tutorial3-genome-lastz \
    --scaffoldlist galGal4 allMis2 \
    --genome-base-path ./ \
    --probefile /nfs/data1/uce-probe-sets/uce-5k-probe-set/uce-5k-probes.fasta \
    --cores 12
```

The program will create some output that looks like:

```
Running against galGal4.2bit
Running with the --huge option.  Chunking files into 10000000 bp...
/tmp/tmptXup1D.lastz

< .. snip .. >

Cleaning up the chunked files...
Cleaning /home/bcf/tmp/uce-genome/tutorial3-genome-lastz/uce-5k-probes.fasta_v_
↪galGal4.lastz
Creating galGal4 table
Inserting data to galGal4 table

Running against allMis2.2bit
Running with the --huge option.  Chunking files into 10000000 bp...
Running the targets against 109 queries...
/tmp/tmpuuHOpS.fasta

< .. snip .. >

Cleaning up the chunked files...
Cleaning /home/bcf/tmp/uce-genome/tutorial3-genome-lastz/uce-5k-probes.fasta_v_
↪allMis2.lastz
Creating allMis2 table
Inserting data to allMis2 table
```

The directory structure should now look like this:

```
uce-genome
+-- allMis2
│   +-- allMis2.2bit
│   +-- sizes.tab
+-- galGal4
│   +-- galGal4.2bit
│   +-- sizes.tab
+-- tutorial3-genome-lastz
│   +-- uce-5k-probes.fasta_v_allMis2.lastz.clean
│   +-- uce-5k-probes.fasta_v_galGal4.lastz.clean
+-- tutorial3.sqlite
+-- uce-5k-probes.fasta
```

Extracting FASTA sequence matching UCE loci from genome sequences

Once you have run the search, you can extract the loci identified during the search from each respective genome sequence plus some sequence flanking each locus (at a distance you can specify). Before you do that, however, you need to create a configuration file that tells the program where to find each genome. In the case above, that file would look like so (the full path to my working directory is */home/bcf/tmp/uce-genome*):

```
[scaffolds]
galGal4:/home/bcf/tmp/uce-genome/galGal4/galGal4.2bit
allMis2:/home/bcf/tmp/uce-genome/allMis2/allMis2.2bit
```

Attention: Be careful to make sure that your capitalization is consistent with your file names.

With that file created as *genomes.conf*, our directory structure looks like:

```
uce-genome
+── allMis2
│   +── allMis2.2bit
│   +── sizes.tab
+── galGal4
│   +── galGal4.2bit
│   +── sizes.tab
+── genomes.conf
+── tutorial3-genome-lastz
│   +── uce-5k-probes.fasta_v_allMis2.lastz.clean
│   +── uce-5k-probes.fasta_v_galGal4.lastz.clean
+── tutorial3.sqlite
+── uce-5k-probes.fasta
```

Now, we can extract FASTA data from each genome for each UCE locus. To do this, we need to input the path to the *lastz* files from above, the path to the *conf* file we just created, the amount of flanking sequence (to each side) that we would like to slice, a name pattern, matching the lastz files that we would like to use, and the name of the output directory we want to create:

```
phyluce_probe_slice_sequence_from_genomes \
    --lastz tutorial3-genome-lastz \
    --conf genomes.conf \
    --flank 500 \
    --name-pattern "uce-5k-probes.fasta_v_{}.lastz.clean" \
    --output tutorial-genome-fasta
```

You should see output similar to:

```
2016-06-01 16:02:18,907 - Phyluce - INFO - =================== Starting Phyluce:␣
↪Slice Sequence ===================
2016-06-01 16:02:18,908 - Phyluce - INFO - ------------------- Working on galGal4␣
↪genome -------------------
2016-06-01 16:02:18,908 - Phyluce - INFO - Reading galGal4 genome
2016-06-01 16:02:28,036 - Phyluce - INFO - galGal4: 4966 uces, 35 dupes, 4931 non-
↪dupes, 2 orient drop, 40 length drop, 4889 written
2016-06-01 16:02:28,036 - Phyluce - INFO - ------------------- Working on allMis2␣
↪genome -------------------
2016-06-01 16:02:28,037 - Phyluce - INFO - Reading allMis2 genome
2016-06-01 16:02:37,230 - Phyluce - INFO - allMis2: 4830 uces, 7 dupes, 4823 non-
↪dupes, 2 orient drop, 3 length drop, 4818 written
```

And, your directory structure should look like:

```
uce-genome
+── allMis2
```

(continues on next page)

(continued from previous page)

```
│   +── allMis2.2bit
│   +── sizes.tab
+── galGal4
│   +── galGal4.2bit
│   +── sizes.tab
+── genomes.conf
+── tutorial3-genome-lastz
│   +── uce-5k-probes.fasta_v_allMis2.lastz.clean
│   +── uce-5k-probes.fasta_v_galGal4.lastz.clean
+── tutorial3.sqlite
+── tutorial-genome-fasta
│   +── allmis2.fasta
│   +── galgal4.fasta
+── uce-5k-probes.fasta
```

The UCE contig sequence are in each respective file within *tutorial-genome-fasta*.

Using the extracted sequences in downstream analyses

The easiest way for you to use the extracted sequences is to symlink them into an appropriate *contigs* folder that resulted from a PHYLUCE assembly process and then proceed with the *Extracting UCE loci* procedure.

For more information on the structure of this folder, look at the *Assemble the data* section of *Tutorial I: UCE Phylogenomics* for more information.

Attention: Although we have already extracted the UCE loci from each genome sequence and even though it seems redundant to go back through the *Extracting UCE loci* process, this is the best path forward.

3.3.4 Tutorial IV: Identifying UCE Loci and Designing Baits To Target Them

The first few tutorials have given you a feel for how to perform phylogenetic/phylogeographic analyses using existing probe sets, new data, and genomes. However, what if you are working on a set of organisms without a probe set targeting conserved loci? How do you identify those loci and design baits to target them? This Tutorial shows you how to do that.

In the examples below, we'll follow a UCE identification and probe design workflow using data from Coleoptera (beetles). Although you can follow the entire tutorial from beginning to end, I've also made the BAM files containing mapped reads available, which lets you skip the computationally exepensive step of performing read simulation and alignment.

Starting directory structure

To keep things clear, we're going to assume you are working in some directory, which I'll call *uce-coleoptera*. We'll be working from the top of this directory in the steps below:

```
uce-coleoptera
```

Data download and preparation

Download the genomes

Attention: You do not neccessarily need to do this as part of this tutorial for UCE identification and probe design - If you only want to follow the steps for locus identification (skipping probe design and in-silico testing), you can simply download the prepared FASTQ/BAM files from figshare.

Make a directory to hold the genome sequences:

```
> mkdir genomes
> cd genomes
```

Now, get the genome sequences:

```
# Anoplophora glabripennis (Asian longhorned beetle)
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/390/285/GCA_000390285.1_Agla_1.
↪0/GCA_000390285.1_Agla_1.0_genomic.fna.gz

# Agrilus planipennis (emerald ash borer)
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/699/045/GCA_000699045.1_Apla_1.
↪0/GCA_000699045.1_Apla_1.0_genomic.fna.gz

# Leptinotarsa decemlineata (Colorado potato beetle)
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/500/325/GCA_000500325.1_Ldec_1.
↪5/GCA_000500325.1_Ldec_1.5_genomic.fna.gz

# Onthophagus taurus (beetles)
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/648/695/GCA_000648695.1_Otau_1.
↪0/GCA_000648695.1_Otau_1.0_genomic.fna.gz

# Dendroctonus ponderosae (mountain pine beetle)
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/355/655/GCA_000355655.1_
↪DendPond_male_1.0/GCA_000355655.1_DendPond_male_1.0_genomic.fna.gz

# Tribolium castaneum (red flour beetle)
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/002/335/GCA_000002335.2_Tcas_3.
↪0/GCA_000002335.2_Tcas_3.0_genomic.fna.gz

# Mengenilla moldrzyki (twisted-wing parasites) [Outgroup]
> wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/281/935/GCA_000281935.1_Memo_1.
↪0/GCA_000281935.1_Memo_1.0_genomic.fna.gz
```

You need to unzip all of the genome sequences

```
> gunzip *
```

Finally, your directory structure should look something like:

```
uce-coleoptera
+-- genomes
    +-- GCA_000002335.2_Tcas_3.0_genomic.fna
    +-- GCA_000281935.1_Memo_1.0_genomic.fna
    +-- GCA_000355655.1_DendPond_male_1.0_genomic.fna
    +-- GCA_000390285.1_Agla_1.0_genomic.fna
```

(continues on next page)

(continued from previous page)

```
    +—— GCA_000500325.1_Ldec_1.5_genomic.fna
    +—— GCA_000648695.1_Otau_1.0_genomic.fna
    +—— GCA_000699045.1_Apla_1.0_genomic.fna
```

Cleanup the genome sequences

Attention: You do not neccessarily need to do this as part of this tutorial for UCE identification and probe design - If you only want to follow the steps for locus identification (skipping probe design and in-silico testing), you can simply download the prepared FASTQ/BAM files from figshare.

When you get genome sequences from NCBI, the FASTA headers of most scaffold/contigs contain a lot of extra cruft that can cause problems with some of the steps in the UCE identification and probe design workflow. I usually remove the extra stuff, maintaining only the accession number information for each contig / scaffold. To do that, I use a little python script that looks like the following:

```
from Bio import SeqIO
with open("Name_of_Genome_File.fna", "rU") as infile:
with open("outfileName.fasta", "w") as outf:
    for seq in SeqIO.parse(infile, 'fasta'):
        seq.name = ""
        seq.description = ""
        outf.write(seq.format('fasta'))
```

For these genome assemblies, the important bits are in the FASTA header just before the space in the name. The code above basically keeps this information before the space and discards the remaining FASTA header information.

In the case of the genomes above, here are the commands I ran (note also that this creates an output file with a different name from the input file):

```
from Bio import SeqIO

# Anoplophora glabripennis (Asian longhorned beetle)
with open("GCA_000390285.1_Agla_1.0_genomic.fna", "rU") as infile:
    with open("anoGla1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
            seq.description = ""
            outf.write(seq.format('fasta'))

# Agrilus planipennis (emerald ash borer)
with open("GCA_000699045.1_Apla_1.0_genomic.fna", "rU") as infile:
    with open("agrPla1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
            seq.description = ""
            outf.write(seq.format('fasta'))

# Dendroctonus ponderosae (mountain pine beetle)
with open("GCA_000355655.1_DendPond_male_1.0_genomic.fna", "rU") as infile:
    with open("denPon1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
```

(continues on next page)

(continued from previous page)

```
            seq.description = ""
            outf.write(seq.format('fasta'))

# Leptinotarsa decemlineata (Colorado potato beetle)
with open("GCA_000500325.1_Ldec_1.5_genomic.fna", "rU") as infile:
    with open("lepDec1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
            seq.description = ""
            outf.write(seq.format('fasta'))

# Mengenilla moldrzyki (twisted-wing parasites) [Outgroup]
with open("GCA_000281935.1_Memo_1.0_genomic.fna", "rU") as infile:
    with open("menMol1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
            seq.description = ""
            outf.write(seq.format('fasta'))

# Onthophagus taurus (beetles)
with open("GCA_000648695.1_Otau_1.0_genomic.fna", "rU") as infile:
    with open("ontTau1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
            seq.description = ""
            outf.write(seq.format('fasta'))

# Tribolium castaneum (red flour beetle)
with open("GCA_000002335.2_Tcas_3.0_genomic.fna", "rU") as infile:
    with open("triCas1.fasta", "w") as outf:
        for seq in SeqIO.parse(infile, 'fasta'):
            seq.name = ""
            seq.description = ""
            outf.write(seq.format('fasta'))
```

Now, you can remove the original files downloaded from NCBI:

```
rm *.fna
```

And, your directory structure should look something like:

```
uce-coleoptera
+-- genomes
    +-- anoGla1.fasta
    +-- agrPla1.fasta
    +-- denPon1.fasta
    +-- lepDec1.fasta
    +-- menMol1.fasta
    +-- ontTau1.fasta
    +-- triCas1.fasta
```

Put genomes in their own directories

Because of some historical reasons (and how I organize our lab data), each genome sequence needs to be in its own directory. We can do that pretty easily by running:

```
> cd uce-coleoptera
> cd genomes
> for critter in *; do mkdir ${critter%.*}; mv $critter ${critter%.*}; done
```

Now the directory structure looks like:

```
uce-coleoptera
+-- genomes
    +-- agrPla1
    |   +-- agrPla1.fasta
    +-- anoGla1
    |   +-- anoGla1.fasta
    +-- denPon1
    |   +-- denPon1.fasta
    +-- lepDec1
    |   +-- lepDec1.fasta
    +-- menMol1
    |   +-- menMol1.fasta
    +-- ontTau1
    |   +-- ontTau1.fasta
    +-- triCas1
        +-- triCas1.fasta
```

Convert genomes to 2bit format

Later during the workflow, we're going to need to have our genomes in *2bit* format, which is a binary format for genomic data that is easier and faster to work with than FASTA format. We'll use a BASH script to convert all of the sequences, above, to *2bit* format:

```
> cd uce-coleoptera
> cd genomes
> for critter in *; do faToTwoBit $critter/$critter.fasta $critter/${critter%.*}.2bit;
↪ done
```

Now the directory structure looks like:

```
uce-coleoptera
+-- genomes
    +-- agrPla1
    |   +-- agrPla1.2bit
    |   +-- agrPla1.fasta
    +-- anoGla1
    |   +-- anoGla1.2bit
    |   +-- anoGla1.fasta
    +-- denPon1
    |   +-- denPon1.2bit
    |   +-- denPon1.fasta
    +-- lepDec1
    |   +-- lepDec1.2bit
    |   +-- lepDec1.fasta
    +-- menMol1
    |   +-- menMol1.2bit
    |   +-- menMol1.fasta
    +-- ontTau1
    |   +-- ontTau1.2bit
```

(continues on next page)

(continued from previous page)

```
    │    +── ontTau1.fasta
    +── triCas1
         +── triCas1.2bit
         +── triCas1.fasta
```

Simulate reads from genomes

Attention: You do not neccessarily need to take this step as part of the tutorial - you can simply download prepared, simulated FASTQ files from figshare.

In order to locate UCE loci across a selection of different genomes, we're going to align reads from each taxon, above, to a reference genome sequence (the "base" genome sequence) using a permissive raw read aligner. You can use reads from low-coverage, genome scans or you can use reads simulated from particular genomes. In this tutorial, we're going to use this latter approach and simulate reads (without sequencing error) from the genomes that we will align to the base genome. To accomplish this, we'll use art, which is a robust read simulator that is reasonably flexible.

Because we're using simulated reads to locate UCE loci, we want to turn off the built-in feature of *art* that adds some sequencing error to simulated reads. This results in a general form of the call to *art* that looks like:

```
> art_illumina \
    --paired \
    --in ~/path/to/input/genome.fasta \
    --out prefix-of-output-file \
    --len 100 \
    --fcov 2 \
    --mflen 200 \
    --sdev 150 \
    -ir 0.0 -ir2 0.0 -dr 0.0 -dr2 0.0 -qs 100 -qs2 100 -na
```

This simulates reads from the *–in genome.fasta* that are 100 bp in length, cover the genome randomly to roughly 2X, have an insert size of 200 bp, and have an inserts size standard deviation of 150. The last line turns off the simulation of sequencing error in each of these reads and also turns off the creation of an alignment file showing where the reads came from in the genome sequence.

In the case of the Coleoptera genomes you downloaded, here are the commands I ran to simulate the read data that we will use in the next step (note also that this creates an output file with a different name from the input file). First, create a directory to hold the simulated read data:

```
> cd uce-coleoptera
> mkdir reads
> cd reads
```

Then, simulate the reads using *art*:

```
> art_illumina \
    --paired \
    --in ../genomes/agrPla1/agrPla1.fasta \
    --out agrPla1-pe100-reads \
    --len 100 --fcov 2 --mflen 200 --sdev 150 -ir 0.0 -ir2 0.0 -dr 0.0 -dr2 0.0 -qs␣
↪100 -qs2 100 -na

> art_illumina \
```

(continues on next page)

(continued from previous page)

```
    --paired \
    --in ../genomes/anoGla1/anoGla1.fasta \
    --out anoGla1-pe100-reads \
    --len 100 --fcov 2 --mflen 200 --sdev 150 -ir 0.0 -ir2 0.0 -dr 0.0 -dr2 0.0 -qs␣
↪100 -qs2 100 -na

> art_illumina \
    --paired \
    --in ../genomes/denPon1/denPon1.fasta \
    --out denPon1-pe100-reads \
    --len 100 --fcov 2 --mflen 200 --sdev 150 -ir 0.0 -ir2 0.0 -dr 0.0 -dr2 0.0 -qs␣
↪100 -qs2 100 -na

> art_illumina \
    --paired \
    --in ../genomes/lepDec1/lepDec1.fasta \
    --out lepDec1-pe100-reads \
    --len 100 --fcov 2 --mflen 200 --sdev 150 -ir 0.0 -ir2 0.0 -dr 0.0 -dr2 0.0 -qs␣
↪100 -qs2 100 -na

> art_illumina \
    --paired \
    --in ../genomes/ontTau1/ontTau1.fasta \
    --out ontTau1-pe100-reads \
    --len 100 --fcov 2 --mflen 200 --sdev 150 -ir 0.0 -ir2 0.0 -dr 0.0 -dr2 0.0 -qs␣
↪100 -qs2 100 -na
```

Now, you should see 2 read files for each taxon. We are going to "break" the pairs by merging the read information together, then we are going to zip the resulting file that contains the *R1* and *R2* data. We can accomplish this pretty easily using a quick BASH script:

Attention: Note that we've dropped menMol1 from the read simulation process. This is largely because it is an outgroup to beetles. We'll use it later, when we're performing *in silico* tests of the UCE bait set.

```
for critter in agrPla1 anoGla1 denPon1 lepDec1 ontTau1;
    do
        echo "working on $critter";
        touch $critter-pe100-reads.fq;
        cat $critter-pe100-reads1.fq > $critter-pe100-reads.fq;
        cat $critter-pe100-reads2.fq >> $critter-pe100-reads.fq;
        rm $critter-pe100-reads1.fq;
        rm $critter-pe100-reads2.fq;
        gzip $critter-pe100-reads.fq;
    done;
```

If we take a look at our directory structure, it should look like:

```
uce-coleoptera
+── genomes (collapsed)
+── reads
    +── anoGla1-pe100-reads.fq.gz
    +── agrPla1-pe100-reads.fq.gz
    +── denPon1-pe100-reads.fq.gz
    +── lepDec1-pe100-reads.fq.gz
```

(continues on next page)

(continued from previous page)

```
    +-- ontTau1-pe100-reads.fq.gz
```

Read alignment to the base genome

Attention: You do not neccessarily need to run this step as part of the tutorial - you can simply download the prepared, BAM files from figshare.

Because we also provide the BAM files created below, you can choose to just start with the BAM files in the *Conserved locus identification* section.

Prepare the base genome

Now that we have read data representing each of our exemplar taxa, we need to align these reads to the "base" genome sequence, in this case the genome sequence of *Tribolium castaneum* (aka *triCas1*). We selected this assembly as the "base" genome because of it's age (i.e., better-assembled) and level of annotation.

We will perform the read alignments to *triCas1* using the permissive read aligner, stampy, which works well when aligning sequences to a divergent reference sequence. Hoever, before running the alignments, we need to prepare the base genome. And, before we do that, let's create a direcetory to work in:

```
> cd uce-coleoptera
> mkdir base
> cd base
```

Now, let's copy the base genome to this directory (for simplicity):

```
> cp ../genomes/triCas1.fasta ./
```

If we take a look at our directory structure, it now looks like:

```
uce-coleoptera
+-- base
|   +-- triCas1.fasta
+-- genomes
|   +-- agrPla1
|   |   +-- agrPla1.2bit
|   |   +-- agrPla1.fasta
|   +-- anoGla1
|   |   +-- anoGla1.2bit
|   |   +-- anoGla1.fasta
|   +-- denPon1
|   |   +-- denPon1.2bit
|   |   +-- denPon1.fasta
|   +-- lepDec1
|   |   +-- lepDec1.2bit
|   |   +-- lepDec1.fasta
|   +-- menMol1
|   |   +-- menMol1.2bit
|   |   +-- menMol1.fasta
|   +-- ontTau1
|   |   +-- ontTau1.2bit
|   |   +-- ontTau1.fasta
```

(continues on next page)

(continued from previous page)

```
│   +── triCas1
│       +── triCas1.2bit
│       +── triCas1.fasta
+── reads
    +── anoGla1-pe100-reads
    +── agrPla1-pe100-reads
    +── denPon1-pe100-reads
    +── lepDec1-pe100-reads
    +── ontTau1-pe100-reads
```

Now, we need to run the commands to prepare the *triCas1* genome for use with stampy:

```
> cd uce-coleoptera
> cd base
> stampy.py --species="tribolium-castaneum" --assembly="triCas1" -G triCas1 triCas1.
↪fasta
> stampy.py -g triCas1 -H triCas1
```

If we look at our directory structure, it should look like:

```
uce-coleoptera
+── base
│   +── triCas1.fasta
│   +── triCas1.sthash
│   +── triCas1.stidx
+── genomes (collapsed)
+── reads
    +── anoGla1-pe100-reads
    +── agrPla1-pe100-reads
    +── denPon1-pe100-reads
    +── lepDec1-pe100-reads
    +── ontTau1-pe100-reads
```

Align reads to the base genome

Attention: You do not neccessarily need to run this step as part of the tutorial - you can simply download the prepared, BAM files from figshare.

Because we also provide the BAM files created below, you can choose to just start with the BAM files in the *Conserved locus identification* section.

Now that we've prepared our base genome, we need to perform the actual alignment of the simulated reads to the base genome. And, before we do that, let's create a directory to hold the resulting alignments:

```
> cd uce-coleoptera
> mkdir alignments
```

Our resulting directory structure should now look like:

```
uce-coleoptera
+── alignments
+── base
│   +── triCas1.fasta
```

(continues on next page)

(continued from previous page)

```
│   +── triCas1.sthash
│   +── triCas1.stidx
+── genomes (collapsed)
+── reads
    +── anoGla1-pe100-reads
    +── agrPla1-pe100-reads
    +── denPon1-pe100-reads
    +── lepDec1-pe100-reads
    +── ontTau1-pe100-reads
```

Now, we need to perform the alignments on a taxon-by-taxon basis (and/or you can run these in parallel using HPC). To do this easily (and on a local computer) we can use a BASH script to run the alignments serially:

Warning: Note that I am using 16 physical CPU cores (*$cores*) to do this work. You need to use the number of physical cores available on *your* machine.

```
export cores=16
export base=triCas1
export base_dir=$HOME/uce-coleoptera/alignments
for critter in agrPla1 anoGla1 denPon1 lepDec1 ontTau1;
    do
        export reads=$critter-pe100-reads.fq.gz;
        mkdir -p $base_dir/$critter;
        cd $base_dir/$critter;
        stampy.py --maxbasequal 93 -g ../../base/$base -h ../../base/$base \
        --substitutionrate=0.05 -t$cores --insertsize=400 -M \
        ../../reads/$reads | samtools view -Sb - > $critter-to-$base.bam;
    done;
```

This code basically loops over each exemplar genomes, aligns the reads to the base genome sequence, and converts the resulting output to BAM format (which is a binary, compressed version of SAM format).

When the alignments have completed, your directory structure should look something like:

```
uce-coleoptera
+── alignments
│    +── anoGla1
│    │   +── anoGla1-to-triCas1.bam
│    +── agrPla1
│    │   +── agrPla1-to-triCas1.bam
│    +── denPon1
│    │   +── denPon1-to-triCas1.bam
│    +── lepDec1
│    │   +── lepDec1-to-triCas1.bam
│    +── ontTau1
│        +── ontTau1-to-triCas1.bam
│
+── base
│   +── triCas1.fasta
│   +── triCas1.sthash
│   +── triCas1.stidx
+── genomes (collapsed)
+── reads
    +── anoGla1-pe100-reads
    +── agrPla1-pe100-reads
```

(continues on next page)

(continued from previous page)

```
    +-- denPon1-pe100-reads
    +-- lepDec1-pe100-reads
    +-- ontTau1-pe100-reads
```

Now, these BAM files are pretty large because they contain all mapped as well as all unmapped reads. We want to remove those unmapped reads, which will also reduce file-size. We can do that using samtools *view* and a BASH script. We're also going to create a directory named *all* and symlink all of the reduced BAM files to this directory.

Warning: The script, as-written, removes the BAM files containing both mapped and unmapped reads. If you don't want to do this, remove the *rm $critter/$critter-to-triCas1.bam;* line.

```
> cd uce-coleoptera
> cd alignments
> mkdir all
> for critter in agrPla1 anoGla1 denPon1 lepDec1 ontTau1;
    do
        samtools view -h -F 4 -b $critter/$critter-to-triCas1.bam > $critter/$critter-
↪to-triCas1-MAPPING.bam;
        rm $critter/$critter-to-triCas1.bam;
        ln -s ../$critter/$critter-to-triCas1-MAPPING.bam all/$critter-to-triCas1-
↪MAPPING.bam;
    done;
```

Now, your directory structure should look something like:

```
uce-coleoptera
+-- alignments
|    +-- anoGla1
|    |   +-- anoGla1-to-triCas1-MAPPING.bam
|    +-- all
|    |   +-- agrPla1-to-triCas1-MAPPING.bam -> ../agrPla1/agrPla1-to-triCas1-MAPPING.
↪bam
|    |   +-- anoGla1-to-triCas1-MAPPING.bam -> ../anoGla1/anoGla1-to-triCas1-MAPPING.
↪bam
|    |   +-- denPon1-to-triCas1-MAPPING.bam -> ../denPon1/denPon1-to-triCas1-MAPPING.
↪bam
|    |   +-- lepDec1-to-triCas1-MAPPING.bam -> ../lepDec1/lepDec1-to-triCas1-MAPPING.
↪bam
|    |   +-- ontTau1-to-triCas1-MAPPING.bam -> ../ontTau1/ontTau1-to-triCas1-MAPPING.
↪bam
|    +-- agrPla1
|    |   +-- agrPla1-to-triCas1-MAPPING.bam
|    +-- denPon1
|    |   +-- denPon1-to-triCas1-MAPPING.bam
|    +-- lepDec1
|    |   +-- lepDec1-to-triCas1-MAPPING.bam
|    +-- ontTau1
|        +-- ontTau1-to-triCas1-MAPPING.bam
+-- base
|   +-- triCas1.fasta
|   +-- triCas1.sthash
|   +-- triCas1.stidx
+-- genomes (collapsed)
+-- reads
```

(continues on next page)

(continued from previous page)

```
    +-- anoGla1-pe100-reads
    +-- agrPla1-pe100-reads
    +-- denPon1-pe100-reads
    +-- lepDec1-pe100-reads
    +-- ontTau1-pe100-reads
```

These *-MAPPING.bam files are available from figshare.

What it all means

Basically, because we've now mapped simulated (or actual) sequence data from several exemplar taxa to a closely-related "base" genome sequence, we've essentially identified putatively orthologous loci shared between the exemplar taxa and the base taxon. These conserved regions are where the simulated (or actual) sequene data mapped with a sequence divergence of < 5%.

Now, we need to filter this large number of conserved regions to remove things like repetitive regions, but also to find *which* loci are shared among *all* exemplar taxa - not simply a single exemplar and the base taxon.

Conserved locus identification

You can download the alignment data generated in the steps above from figshare for use in subsequent steps, rather than generating them youself (thus, saving you time).

Attention: If you are starting the tutorial at this position after downloading the *-MAPPING.bam files from figshare, then you will need to create a directory to work in named *uce-coleoptera* and then place all of the **-MAPPING.bam* files in a subdirectory of *uce-coleoptera* names *alignments/all*. Your resulting directory structure should look like:

```
uce-coleoptera
+-- alignments
|   +-- all
|       +-- agrPla1-to-triCas1-MAPPING.bam -> ../agrPla1/agrPla1-to-triCas1-MAPPING.
↪bam
|       +-- anoGla1-to-triCas1-MAPPING.bam -> ../anoGla1/anoGla1-to-triCas1-MAPPING.
↪bam
|       +-- denPon1-to-triCas1-MAPPING.bam -> ../denPon1/denPon1-to-triCas1-MAPPING.
↪bam
|       +-- lepDec1-to-triCas1-MAPPING.bam -> ../lepDec1/lepDec1-to-triCas1-MAPPING.
↪bam
|       +-- ontTau1-to-triCas1-MAPPING.bam -> ../ontTau1/ontTau1-to-triCas1-MAPPING.
↪bam
+-- genomes (collapsed)
```

If you want to go beyond conserved locus identification and design probes from the target taxa, you will also need to download the appropriate genomes. See the *Data download and preparation* section.

Convert BAMS to BEDS

In the steps above, we have generated BAM files representing reads that stampy has mapped to the base genome. Those reads that map align to putatively conserved sequence regions (that we need to filter), and these alignments of mapping reads should reside in our *alignments/all* directory.

To begin the filtering process, we're going to convert each BAM file to BED format, which is an interval-based format that is easy and fast to manipulate with a software suite known as bedtools. But before we do that, we are doing to create a *bed* directory to hold all of these BED format files.

```
> cd uce-coleoptera

# make a directory to hold the BED data
> mkdir bed
> cd bed

# now, convert our *-MAPPING.bam files to BED format
> for i in ../alignments/all/*.bam; do echo $i; bedtools bamtobed -i $i -bed12 >␣
↪`basename $i`.bed; done
```

Your directory structure should look something like:

```
uce-coleoptera
+—— alignments
│   +—— all
│       +—— agrPla1-to-triCas1-MAPPING.bam
│       +—— anoGla1-to-triCas1-MAPPING.bam
│       +—— denPon1-to-triCas1-MAPPING.bam
│       +—— lepDec1-to-triCas1-MAPPING.bam
│       +—— ontTau1-to-triCas1-MAPPING.bam
+—— bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.bed
│   +—— denPon1-to-triCas1-MAPPING.bam.bed
│   +—— lepDec1-to-triCas1-MAPPING.bam.bed
│   +—— ontTau1-to-triCas1-MAPPING.bam.bed
+—— genomes (collapsed)
```

Sort the converted BEDs

Before moving forward with the *merge* command, below, we need to sort the resulting BED files, which orders each lines of data in the BED file by chromosome/scaffold/contig and position along that chromosome/scaffold/contig. Again, we can do this with some bash scripting:

```
> cd uce-coleoptera
> cd bed
> for i in *.bed; do echo $i; bedtools sort -i $i > ${i%.*}.sort.bed; done
```

Your directory structure should look something like the following (note that I have collapsed the directory listing for *all*):

```
uce-coleoptera
+—— alignments
│   +—— all (collapsed)
+—— bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.sort.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.sort.bed
│   +—— denPon1-to-triCas1-MAPPING.bam.bed
│   +—— denPon1-to-triCas1-MAPPING.bam.sort.bed
│   +—— lepDec1-to-triCas1-MAPPING.bam.bed
```

(continues on next page)

(continued from previous page)

```
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.bed
+── genomes (collapsed)
```

Merge overlapping or nearly-overlapping intervals

Because there may be small gaps between proximate regions of conservation (which may result because we're using data that are either from low-coverage, simulated reads or low-coverage *actual* reads) we need to merge together putative regions of conservation that are proximate. Luckily bedtools has a handy tool to do that - the *merge* function.

```
> cd uce-coleoptera
> cd bed
> for i in *.bam.sort.bed; do echo $i; bedtools merge -i $i > ${i%.*}.merge.bed; done
```

Your directory structure should look something like the following (note that I have collapsed the directory listing for *all*):

```
uce-coleoptera
+── alignments
│   +── all (collapsed)
+── bed
│   +── agrPla1-to-triCas1-MAPPING.bam.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── denPon1-to-triCas1-MAPPING.bam.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed
+── genomes (collapsed)
```

To get some idea of the total number of merged, putatively conserved regions in each exemplar taxon that are shared with the base genome, we can simply loop over the files and count the number of lines in each:

```
> cd uce-coleoptera
> cd bed

> for i in *.bam.sort.merge.bed; do wc -l $i; done
19810 agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed
48350 anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed
21390 denPon1-to-triCas1-MAPPING.bam.sort.merge.bed
33144 lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed
25188 ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed
```

Remove repetitive intervals

At this point, we've mapped reads to the base genome, kept those regions where reads mapped, converted that to a BED, and merged intervals that are very close to one another.

What we have not done is remove any putatively conserved intervals shared between exemplar taxa and the base genome that are repetitive regions. To do this, we're going to run a python program over all of the BED files for each exemplar taxon. This program will look at the intervals shared between the exemplar taxon and the base genome and it will remove intervals where the base genome is shorter than 80 bp and where > 25 % of the base genome is masked.

```
> cd uce-coleoptera
> cd bed

> for i in *.sort.merge.bed;
    do
        phyluce_probe_strip_masked_loci_from_set \
            --bed $i \
            --twobit ../genomes/triCas1/triCas1.2bit \
            --output ${i%.*}.strip.bed \
            --filter-mask 0.25 \
            --min-length 80
    done;

Screened 19810 sequences from agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed. ␣
↪Filtered 3113 with > 25.0% masked bases or > 0 N-bases or < 80 length. Kept 16697.
Screened 48350 sequences from anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed. ␣
↪Filtered 13226 with > 25.0% masked bases or > 0 N-bases or < 80 length. Kept 35124.
Screened 21390 sequences from denPon1-to-triCas1-MAPPING.bam.sort.merge.bed. ␣
↪Filtered 3008 with > 25.0% masked bases or > 0 N-bases or < 80 length. Kept 18382.
Screened 33144 sequences from lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed. ␣
↪Filtered 6585 with > 25.0% masked bases or > 0 N-bases or < 80 length. Kept 26559.
Screened 25188 sequences from ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed. ␣
↪Filtered 6505 with > 25.0% masked bases or > 0 N-bases or < 80 length. Kept 18683.
```

When this finishes, your directory structure should look like:

```
uce-coleoptera
+── alignments
│   +── all (collapsed)
+── bed
│   +── agrPla1-to-triCas1-MAPPING.bam.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── denPon1-to-triCas1-MAPPING.bam.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.bed
```

(continues on next page)

(continued from previous page)

```
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
+── genomes (collapsed)
```

Determining locus presence in multiple genomes

Up to this point, we've been processing each file on a taxon-by-taxon basis, where each taxon had data aligned to the base genome. Now, we need to determine which loci are conserved *across* taxa. To do that, we first need to prepare a configuration file (named *bed-files.conf*) that gives the paths to each of our **.bam.sort.merge.strip.bed* files. That file needs to be in configuration file format, like so:

```
[beds]
agrPla1:agrPla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
anoGla1:anoGla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
denPon1:denPon1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
lepDec1:lepDec1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
ontTau1:ontTau1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
```

The *[beds]* line is the "header" line, and that is followed by each taxon name (on the left) and the name of the BED file we want to process (on the right). You should place this file in the *bed* directory. If you place it elsewhere, you'll need to use full paths on the right hand side.

Your directory structure should now look like (note new *bed-files.conf*)

```
uce-coleoptera
+── alignments
│   +── all (collapsed)
+── bed
│   +── agrPla1-to-triCas1-MAPPING.bam.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── agrPla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── anoGla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── bed-files.conf
│   +── denPon1-to-triCas1-MAPPING.bam.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── denPon1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── lepDec1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── menMol1-to-triCas1-MAPPING.bam.bed
│   +── menMol1-to-triCas1-MAPPING.bam.sort.bed
│   +── menMol1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── menMol1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +── ontTau1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
+── genomes (collapsed)
```

Now, we're going to run the following program, that creates a record of which alignment intervals are shared among taxa. We need to pass the location of the *bed-files.conf* to this program, along with the name of our base genome, and a name for the output database that will be created:

```
> phyluce_probe_get_multi_merge_table \
    --conf bed-files.conf \
    --base-taxon triCas1 \
    --output coleoptera-to-triCas1.sqlite

agrpla1.................
anogla1...................................
denpon1...................
lepdec1...........................
onttau1...................
Creating database
Inserting results
```

The program shows the results of inserting data for each exemplar taxon that we've selected. If we take a look at the table contents (see *The probe.matches.sqlite database* for more instructions on sqlite databases), we see something like the following:

```
sqlite> select * from triCas1 limit 10;
uce         chromo      start       stop        agrpla1     anogla1     denpon1     ␣
↪lepdec1     onttau1
----------  ----------  ----------  ----------  ----------  ----------  ----------  --
↪--------   ----------
1           GG695505.1  532         632         1           1           0           0␣
↪           1
2           GG695547.1  826         926         0           1           0           0␣
↪           0
3           GG695547.1  1121        1221        0           1           0           0␣
↪           0
4           GG695547.1  1293        1393        0           1           0           0␣
↪           0
5           GG694821.1  1002        1102        0           0           0           1␣
↪           0
6           GG695519.1  73          193         0           1           1           0␣
↪           0
7           GG695519.1  222         380         1           0           1           0␣
↪           1
8           GG695519.1  925         1129        1           1           1           0␣
↪           1
9           DS497688.1  17907       18022       0           0           1           0␣
↪           0
10          DS497688.1  19840       19934       0           1           0           0␣
↪           0
```

The first row of this table (which is limited to 10 rows of results by the query although it is 60699 rows long) shows that for *triCas1* contig *GG695505.1*, *agrpla1*, *anogla1*, and *onttau1* have reads that overlap at position 532 to 632.

Determining shared, conserved, loci

Now that we have our table of results, we can run a quick query (using a Python program) against the table to look at results, more generally. The following code queries the database and writes out the number of loci shared by the base taxon (*triCas1*) and 1, 2, 3, 4, and 5 (all) of the exemplar taxa that we've aligned to the base genome. You need to give the program the path to the database created above and the name of the base taxon:

```
> phyluce_probe_query_multi_merge_table \
        --db coleoptera-to-triCas1.sqlite  \
        --base-taxon triCas1

Loci shared by triCas1 + 0 taxa:    60,699.0
Loci shared by triCas1 + 1 taxa:    60,699.0
Loci shared by triCas1 + 2 taxa:    32,431.0
Loci shared by triCas1 + 3 taxa:    15,834.0
Loci shared by triCas1 + 4 taxa:    6,471.0
Loci shared by triCas1 + 5 taxa:    1,822.0
```

The output from this program basically says that, if we are interested in only those loci found in all exemplar taxa that align to the base genome, there are 1,822 of those. Similarly, if we're willing to be a little less strict about things, there are 6,471 conserved loci that are shared by triCas1 and 4 of the exemplar taxa.

Question: How conservative should I be?

Basically, the question boils down to "Should I select only the set of loci shared by all exemplars and the base genome or shoul I be more liberal?". It's also a hard question to answer. In most cases, I'm pretty happy selecting *n-1* or *n-2* where *n* is the total number of exemplar taxa. In the example below, however, we've selected *n* as the "ideal". This is largely because we have so little information about coleopteran genomes - so we want to be pretty darn sure these loci are found in most/all of them.

Now that we have a general sense of the number of conserved loci in each class of sharing across exemplars (e.g. 5 (all), 4, 3, 2, 1), we need to extract those loci that fall within one of these classes. In this case (and as noted in the box, above), we're going to ouput only those conserved loci that we've identified as being shared between the base genome and *all* exemplars. We do that with a slightly different Python script. This script takes the database name, the base genome, the count of exemplar taxa shared across, and the name of an output file as input. The output file will be BED formatted.

```
> phyluce_probe_query_multi_merge_table \
        --db coleoptera-to-triCas1.sqlite \
        --base-taxon triCas1 \
        --output triCas1+5.bed \
        --specific-counts 5

Counter({'anogla1': 1822, 'lepdec1': 1822, 'agrpla1': 1822, 'denpon1': 1822, 'onttau1
↪': 1822})
```

Conserved locus validation

Extract FASTA sequence from base genome for temp bait design

Now that we've indentified conserved sequences shared among the base genome and the exemplar taxa, we need to start designing baits to capture these loci. The first step in this process is to extract FASTA sequences from the base genome that correspond to the loci we've identified. We do that with a Python script that takes as input the BED file we created, above, the *2bit*-formatted base genomes, a length of sequence we want to extract (160 bp), and an output FASTA filename.

Question: Why buffer to 160 bp?

We are extracting FASTA regions of 160 bp because that allows us to place 2 baits right in the center of this region at 3x tiling density which means that standard 120 bp baits will overlap by 40 bp and have 80 bp to each side (total length 160 bp).

To run the code, we use:

```
> phyluce_probe_get_genome_sequences_from_bed \
        --bed triCas1+5.bed  \
        --twobit ../genomes/triCas1/triCas1.2bit \
        --buffer-to 160 \
        --output triCas1+5.fasta

Screened 1822 sequences.  Filtered 7 < 160 bp or with > 25.0% masked bases or > 0 N-
↪bases. Kept 1815.
```

That should produce a fasta file whose contents look similar to:

```
>slice_0 |DS497688.1:249724-250111
AAAATCAAAGTCGAATACAAAGGCGAATCTAAGACTTTCTATCCTGAAGAGATCAGTTCC
ATGGTacttacaaaaatgaaggaaacTGCCGAAGCCTATTTAGGCAAATCGGTCACAAAT
GCCGTTATCACCGTACCAGCCTATTTCAACGATTCGCAAAGGCAGGCAACTAAAGATGCC
GGTACTATTTCCGGCTTGCAAGTTTTGCGTATTATTAACGAACCTACGGCTGCTGCCATT
GCCTACGGTTTGGATAAGAAGGGAACTGGGGAACGTAATGTCTTGATTTTTGATCTGGGT
GGTGGTACTTTTGATGTGAGCATTTTGACCATTGAGGATGGCATTTTCGAGGTCAAGTCC
ACCGCTGGTGATACGCATTTGGGTGGC
>slice_1 |DS497688.1:250513-250673
CCTGATGAGGCTGTTGCCTATGGAGCTGCCGTCCAAGCCGCCATTTTGCACGGTGATAAG
TCGGAAGAGGTTCAAGATTTGCTACTTTTGGACGTTACTCCACTTTCATTGGGTATTGAA
ACAGCAGGCGGTGTGATGACTGCTTTGATCAAGCGTAACA
>slice_2 |DS497688.1:250682-250991
CAACCAAACAAACGCAAACTTTCACCACCTACTCTGATAACCAACCCGGTGTATTGATCC
AAGTGTACGAAGGCGAACGAGCGATGACTAAAGACAATAACCTTTTGGGTAAATTCGAAT
TGACTGGAATCCCACCGGCACCAAGAGGTGTTCCCCAAATCGAAGTCACCTTTGATATTG
ACGCCAACGGGATTTTGAACGTCACAGCCATCGAGAAGAGCACCAACAAGGAGAACAAAA
TCACCATCACCAATGATAAGGGACGTTTGAGCAAGGAAGATATCGAACGGATGGTCAACG
AAGCCGAGA
```

Design a temporary bait set from the base taxon

Now that we've extracted the appropriate loci from the base genome, we need to design bait sequences targeting these loci. For that, we use a different Python script. This program takes as input the FASTA file we just created, and some design-specific information (*–probe-prefix, –design, –designer*). The design options (*–tiling-density, –two-probes, –overlap*) ensure that we select two baits per locus with 3x tiling that overlap the middle of the targeted locus. Finally, we remove (*–masking, –remove-gc*) potentially problematic baits with >25% repeat content and GC content outside of the range of 30-70% (30 % > GC > 70%).

```
> phyluce_probe_get_tiled_probes \
    --input triCas1+5.fasta \
    --probe-prefix "uce-" \
    --design coleoptera-v1 \
    --designer faircloth \
    --tiling-density 3 \
    --two-probes \
    --overlap middle \
    --masking 0.25 \
    --remove-gc \
    --output triCas1+5.temp.probes
```

(continues on next page)

(continued from previous page)

```
Probes removed for masking (.) / low GC % (G) / ambiguous bases (N):
GGGGGGGGGGGGGGGGGGGGGGGGGGGG


Conserved locus count = 1805
Probe Count = 3602
```

Remove duplicates from our temporary bait set

Because we haven't search for duplicates among our loci and because reducing longer reads to shorter ones (e.g. designing baits from loci) can introduce duplicate baits, we need to screen the resulting bait set for duplicates. To do that, we follow a 2-stage process - first to align all probes to themselves then to use those alignments to remove potentially duplicates baits/loci. First we run a lastz search of all baits to themselves. This program takes as input the temp probes we just designed (as both *–target* and *–query*), relatively low values for *–identity* and *–coverage* to make sure we identify as many duplicates as possible, and the program writes these results to the *–output* file:

```
> phyluce_probe_easy_lastz \
    --target triCas1+5.temp.probes \
    --query triCas1+5.temp.probes \
    --identity 50 --coverage 50 \
    --output triCas1+5.temp.probes-TO-SELF-PROBES.lastz

Started:  Fri Jun 03, 2016  13:57:54
Ended:  Fri Jun 03, 2016  13:57:55
Time for execution:  0.0284410158793 minutes
```

Now that we've run the alignments, we need to screen them alignments and remove the duplicate baits from the bait set. This program takes as input the lastz results from above and the temp-probe file, as well as the probe-prefix that we used during probe design, above. The results are written to a file that is equivalent to the probe file name + *DUPE-SCREENED*, so in this case the output file is named *triCas1+5.temp-DUPE-SCREENED.probes*.

```
> phyluce_probe_remove_duplicate_hits_from_probes_using_lastz \
    --fasta triCas1+5.temp.probes  \
    --lastz triCas1+5.temp.probes-TO-SELF-PROBES.lastz \
    --probe-prefix=uce-

Parsing lastz file...
Screening results...
Screened 3601 fasta sequences.  Filtered 292 duplicates. Kept 3019.
```

Align baits against exemplar genomes

Now that we have a duplicate-free (or putatively duplicate free) set of temporary baits designed from conserved loci in the base genome, we're going to use some in-silico alignments to see if we can locate these loci in the several exemplar genomes.

Attention: For the following analyses, you need genome assemblies for each of the exemplar taxa, formatted as *2bit* files.

We'll use the results of these alignments to design a bait set that includes baits designed from the base genome, but also from the exemplar taxa. This should allow our bait set to work more consistently across broad groups of organisms.

In terms of directory structure, things should look pretty similar to the following:

```
uce-coleoptera
+—— alignments
│   +—— all (collapsed)
+—— bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.sort.bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +—— agrPla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.sort.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +—— anoGla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +—— bed-files.conf
│   +—— coleoptera-to-triCas1.sqlite
│   +—— denPon1-to-triCas1-MAPPING.bam.bed
│   +—— denPon1-to-triCas1-MAPPING.bam.sort.bed
│   +—— denPon1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +—— denPon1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +—— lepDec1-to-triCas1-MAPPING.bam.bed
│   +—— lepDec1-to-triCas1-MAPPING.bam.sort.bed
│   +—— lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +—— lepDec1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +—— ontTau1-to-triCas1-MAPPING.bam.bed
│   +—— ontTau1-to-triCas1-MAPPING.bam.sort.bed
│   +—— ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +—— ontTau1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +—— triCas1+5.bed
│   +—— triCas1+5.bed.missing.matrix
│   +—— triCas1+5.fasta
+—— genomes
    +—— agrPla1
    │   +—— agrPla1.2bit
    │   +—— agrPla1.fasta
    +—— anoGla1
    │   +—— anoGla1.2bit
    │   +—— anoGla1.fasta
    +—— denPon1
    │   +—— denPon1.2bit
    │   +—— denPon1.fasta
    +—— lepDec1
    │   +—— lepDec1.2bit
    │   +—— lepDec1.fasta
    +—— menMol1
    │   +—— menMol1.2bit
    │   +—— menMol1.fasta
    +—— ontTau1
    │   +—— ontTau1.2bit
    │   +—— ontTau1.fasta
    +—— triCas1
        +—— triCas1.2bit
        +—— triCas1.fasta
```

Note that we have all the genomes in their directory, in both FASTA and *2bit* formats. We're also have a new genome sequence in here - that of *menMol1* (Mengenilla moldrzyki [twisted-wing parasites]), which represents the outgroup to Coleoptera. We're adding this taxon because it helps us bridge the base of the tree - e.g. the divergence between the outgroup and the exemplar taxa that we're using to design probes.

Question: What exemplar taxa should I use for bait design?

This is a really hard question to answer. In old, divergent groups with few genomic resources, the answer is usually "all the species" with genomic data. Basically, you want to include exemplars that make the divergence among baits targeting the same loci something >20% or so. That said, even this number is a bit of a guess - no one has systematically tested how "sticky" baits are when they are used to enrich loci across divergent groups. We know they are pretty sticky and in certain cases can enrich loci as much as 35%-40% divergent from the bait sequence. Generally speaking, I try to include exemplar taxa during probe design that bridge the known diversity of a given group… again, in many cases this is hard (or impossible) to know given current data. So, you may have to take a bit of a guess.

So, assuming that you have the appropriate *2bit* files in *uce-coleoptera/genomes*, we are going to align the temporary probes that we've designed to the exemplar genomes, and we're going to run these and subsequent bait design steps in a new directory, named *probe-design*. So:

```
> cd uce-coleoptera
> mkdir probe-design
> cd probe-design
```

Now, you're directory structure should look like:

```
uce-coleoptera
+-- alignments
│   +-- all (collapsed)
+-- bed
│   +-- agrPla1-to-triCas1-MAPPING.bam.bed
│   +-- agrPla1-to-triCas1-MAPPING.bam.sort.bed
│   +-- agrPla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +-- agrPla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +-- anoGla1-to-triCas1-MAPPING.bam.bed
│   +-- anoGla1-to-triCas1-MAPPING.bam.sort.bed
│   +-- anoGla1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +-- anoGla1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +-- bed-files.conf
│   +-- coleoptera-to-triCas1.sqlite
│   +-- denPon1-to-triCas1-MAPPING.bam.bed
│   +-- denPon1-to-triCas1-MAPPING.bam.sort.bed
│   +-- denPon1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +-- denPon1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +-- lepDec1-to-triCas1-MAPPING.bam.bed
│   +-- lepDec1-to-triCas1-MAPPING.bam.sort.bed
│   +-- lepDec1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +-- lepDec1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +-- menMol1-to-triCas1-MAPPING.bam.bed
│   +-- menMol1-to-triCas1-MAPPING.bam.sort.bed
│   +-- menMol1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +-- menMol1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +-- ontTau1-to-triCas1-MAPPING.bam.bed
│   +-- ontTau1-to-triCas1-MAPPING.bam.sort.bed
│   +-- ontTau1-to-triCas1-MAPPING.bam.sort.merge.bed
│   +-- ontTau1-to-triCas1-MAPPING.bam.sort.merge.strip.bed
│   +-- triCas1+5.bed
│   +-- triCas1+5.bed.missing.matrix
│   +-- triCas1+5.fasta
```

(continues on next page)

(continued from previous page)

```
+—— genomes (collapsed)
+—— probe-design
```

We need to align the temporary probe sequences to each genome, which we can do using the following code, which takes as input our temporary probe file, the list of genomes we want to align the probes against, the path to the genomes, the minimum sequence identity to accept a match (on the low end of the spectrum for this step), and number of compute cores to use, and the name of an output database to create and the output directory in which to store the lastz results.

Warning: Note that I am using 16 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

```
> mkdir coleoptera-genome-lastz
> phyluce_probe_run_multiple_lastzs_sqlite \
    --probefile ../bed/triCas1+5.temp-DUPE-SCREENED.probes \
    --scaffoldlist agrPla1 anoGla1 denPon1 lepDec1 ontTau1 triCas1 menMol1 \
    --genome-base-path ../genomes \
    --identity 50 \
    --cores 16 \
    --db triCas1+5+menMol1.sqlite \
    --output coleoptera-genome-lastz

Running against agrPla1.2bit
Running with the --huge option.  Chunking files into 10000000 bp

< ... snip ... >

Cleaning up the chunked files...
Cleaning /nfs/data1/working/bfaircloth-insects/coleoptera/temp/probe-design/
↪coleoptera-genome-lastz/triCas1+5.temp-DUPE-SCREENED.probes_v_menMol1.lastz
Creating menMol1 table
Inserting data to menMol1 table
```

Extract sequence around conserved loci from exemplar genomes

Based on the alignments of the temporary probe set to the exemplar genomes, we need to extract FASTA data from each of the exemplar sequences so that we can design baits targeting the conserved loci in each. This is pretty similar to what we did earlier for the temporary probe set, except that now we're running the extraction across all the exemplar taxa.

Before we begin, we need to make a configuration file with all the genome locations in it (again, as before):

```
[scaffolds]
menMol1:/path/to/uce-coleoptera/genomes/menMol1/menMol1.2bit
agrPla1:/path/to/uce-coleoptera/genomes/agrPla1/agrPla1.2bit
anoGla1:/path/to/uce-coleoptera/genomes/anoGla1/anoGla1.2bit
denPon1:/path/to/uce-coleoptera/genomes/denPon1/denPon1.2bit
lepDec1:/path/to/uce-coleoptera/genomes/lepDec1/lepDec1.2bit
ontTau1:/path/to/uce-coleoptera/genomes/ontTau1/ontTau1.2bit
triCas1:/path/to/uce-coleoptera/genomes/triCas1/triCas1.2bit
```

Using the configuration file, we need to extract the FASTA sequence that we need from each exemplar taxon. Here, we're buffering each locus to 180 bp to give us a little more room to work with during the probe design step. The program takes our config file as input, along with the folder of lastz results created above. The *–name-pattern* argument allows us to match files int the *–lastz* directory, *–probes* is how we buffer the sequence, and we pass the name of an output directory to *–output*:

```
> phyluce_probe_slice_sequence_from_genomes \
    --conf coleoptera-genome.conf \
    --lastz coleoptera-genome-lastz \
    --probes 180 \
    --name-pattern "triCas1+5.temp-DUPE-SCREENED.probes_v_{}.lastz.clean" \
    --output coleoptera-genome-fasta

2016-06-03 15:07:16,642 - Phyluce - INFO - =================== Starting Phyluce:␣
↪Slice Sequence ===================
2016-06-03 15:07:16,644 - Phyluce - INFO - ------------------- Working on menMol1␣
↪genome -------------------
2016-06-03 15:07:16,645 - Phyluce - INFO - Reading menMol1 genome
2016-06-03 15:07:20,221 - Phyluce - INFO - menMol1: 884 uces, 139 dupes, 745 non-
↪dupes, 0 orient drop, 2 length drop, 738 written
2016-06-03 15:07:20,222 - Phyluce - INFO - ------------------- Working on agrPla1␣
↪genome -------------------
2016-06-03 15:07:20,223 - Phyluce - INFO - Reading agrPla1 genome
2016-06-03 15:07:24,759 - Phyluce - INFO - agrPla1: 1410 uces, 184 dupes, 1226 non-
↪dupes, 7 orient drop, 63 length drop, 1156 written
2016-06-03 15:07:24,760 - Phyluce - INFO - ------------------- Working on anoGla1␣
↪genome -------------------
2016-06-03 15:07:24,761 - Phyluce - INFO - Reading anoGla1 genome
2016-06-03 15:07:29,926 - Phyluce - INFO - anoGla1: 1474 uces, 224 dupes, 1250 non-
↪dupes, 6 orient drop, 35 length drop, 1209 written
2016-06-03 15:07:29,926 - Phyluce - INFO - ------------------- Working on denPon1␣
↪genome -------------------
2016-06-03 15:07:29,929 - Phyluce - INFO - Reading denPon1 genome
2016-06-03 15:07:34,472 - Phyluce - INFO - denPon1: 1361 uces, 305 dupes, 1056 non-
↪dupes, 6 orient drop, 30 length drop, 1020 written
2016-06-03 15:07:34,472 - Phyluce - INFO - ------------------- Working on lepDec1␣
↪genome -------------------
2016-06-03 15:07:34,473 - Phyluce - INFO - Reading lepDec1 genome
2016-06-03 15:07:40,020 - Phyluce - INFO - lepDec1: 1436 uces, 259 dupes, 1177 non-
↪dupes, 10 orient drop, 28 length drop, 1139 written
2016-06-03 15:07:40,021 - Phyluce - INFO - ------------------- Working on ontTau1␣
↪genome -------------------
2016-06-03 15:07:40,022 - Phyluce - INFO - Reading ontTau1 genome
2016-06-03 15:07:44,350 - Phyluce - INFO - ontTau1: 1361 uces, 206 dupes, 1155 non-
↪dupes, 9 orient drop, 41 length drop, 1105 written
2016-06-03 15:07:44,350 - Phyluce - INFO - ------------------- Working on triCas1␣
↪genome -------------------
2016-06-03 15:07:44,351 - Phyluce - INFO - Reading triCas1 genome
2016-06-03 15:07:49,499 - Phyluce - INFO - triCas1: 1513 uces, 199 dupes, 1314 non-
↪dupes, 26 orient drop, 46 length drop, 1242 written
```

If we look at out directory structure, it looks something like:

```
uce-coleoptera
+-- alignments (collapsed)
+-- bed (collapsed)
+-- genomes (collapsed)
+-- probe-design
    +-- coleoptera-genome.conf
    +-- coleoptera-genome-fasta
```

(continues on next page)

(continued from previous page)

```
    │   +── agrpla1.fasta
    │   +── anogla1.fasta
    │   +── denpon1.fasta
    │   +── lepdec1.fasta
    │   +── menmol1.fasta
    │   +── onttau1.fasta
    │   +── tricas1.fasta
    +── coleoptera-genome-lastz
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_agrPla1.lastz.clean
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_anoGla1.lastz.clean
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_denPon1.lastz.clean
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_lepDec1.lastz.clean
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_menMol1.lastz.clean
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_ontTau1.lastz.clean
    │   +── triCas1+5.temp-DUPE-SCREENED.probes_v_triCas1.lastz.clean
    +── triCas1+5+menMol1.sqlite
```

The FASTA files we just created are in *uce-coleoptera/coleoptera-genome-fasta*. The output from the program that you see basically shows you how many UCE loci we extracted from each of the exemplar genomes. As expected, the lowest number we located and extracted are from the *menMol1* (outgroup) genome.

If we have a look in one of these FASTA files, it looks like:

```
> less probe-design/coleoptera-genome-fasta/agrpla1.fasta

>slice_0|contig:KL218988.1|slice:301686-301866|uce:uce-500|match:301721-
↪301831|orient:+|probes:1
TCGAACTTCTGGTGCTTGTCACCCTTGATGTCCGCACCGAATTCCTTCACGAGACTGTTT
CTAAAACTTTGGACAAGATGATTGGAGACACAGAAACAGACGAACATCAGAAACGTGTAT
ATCTTCACTTTCTAGAGCATTCATACAAACTTATTACCAGATGTACTCAGCAGCAGCTTT
>slice_1|contig:KL218988.1|slice:319624-319804|uce:uce-501|match:319637-
↪319791|orient:+|probes:2
atgatttttcaaAGGTTACAGCGAAGTCCTCGATTCTACAGCAGATCGAAGAACTAGGA
GAAGAGACTGGCCTGGTGTGCTGTATTTGTCGCGAGGGATACAAGTATCAACCTGCCAAG
GTATTGGGAATTTATACGTTTACAAAGAGGTGCAACGTGGACGAGTTCGAAGCAAAACCA
>slice_2|contig:KL219144.1|slice:184423-184603|uce:uce-503|match:184453-
↪184573|orient:+|probes:1
agttttaaataatcttACCTAAAGAACTAAAATGAAGAAGCATTTCGTCTGCTCGTAAGT
CTTGAGCAATGACATCATCAGCGTAGACACATATTATAGCAAGGCATATAAATAGGTGAA
AGTAGTCTGTTAGATAATTTGCCCAACAAGCTTCCCACAGTCTAAGGGCAACACCTTCGG
>slice_3|contig:KL219144.1|slice:237138-237318|uce:uce-504|match:237150-
↪237306|orient:+|probes:2
CTGTGCAAGAGTGAGTGCCATTGATGCAACACTTGAGCGAGATGATCTAAACCTCCATGG
TGAAAATGAAGAATTTTATATTGAGATTCCCTCGAAGCAACAACCACCTGCCCTGATGTG
CAGCTTGAGTCGTTAAAGAAAAGCCTTAAAGATCTCATTTGGCTTAGATCAACGCTGAAC
```

Find which loci we detect consistently

As before, we want to determine which loci we are detecting consistently across all of the exemplar taxa when doing these in-silico searches. To do that, we'll run another bit of python code. Here, we're working in the *uce-coleoptera/coleoptera-genome-lastz* directory. This program will create a relational database that houses detections of loci in the exemplar taxa. It takes, as input, the folder of FASTAs we just created, the base genome taxon, and a name to use as the output database:

```
> phyluce_probe_get_multi_fasta_table \
    --fastas ../coleoptera-genome-fasta \
    --output multifastas.sqlite \
    --base-taxon triCas1

menmol1.
agrpla1..
anogla1..
denpon1..
lepdec1..
onttau1..
tricas1..
Creating database
Inserting results
```

If we take a look at the table contents in the database (see *The probe.matches.sqlite database* for more instructions on sqlite databases), we see something like the following:

```
locus       menmol1     agrpla1     anogla1     denpon1     lepdec1     onttau1     ␣
↪tricas1
----------  ----------  ----------  ----------  ----------  ----------  ----------  --
↪--------
uce-500     0           1           1           0           1           1           1
uce-501     1           1           1           1           1           1           1
uce-503     1           1           1           1           0           1           1
uce-504     1           1           1           1           1           1           1
uce-505     1           0           1           0           0           1           1
uce-506     1           1           1           1           1           1           1
uce-507     0           1           1           1           1           1           1
uce-508     1           1           1           1           1           0           1
uce-509     1           0           1           1           1           1           1
uce-967     0           0           0           1           1           1           0
```

Which shows our detection of conserved loci in each of the exemplar taxa when we search for them using the temporary probes that we designed from the base genome. As before, we can get some idea of the distribution of hits among exemplar taxa (e.g., are loci detected in "all", n-1 taxa, n-2 taxa, etc.).

```
> phyluce_probe_query_multi_fasta_table \
    --db multifastas.sqlite \
    --base-taxon triCas1

Loci shared by 0 taxa:  1,437.0
Loci shared by 1 taxa:  1,437.0
Loci shared by 2 taxa:  1,355.0
Loci shared by 3 taxa:  1,303.0
Loci shared by 4 taxa:  1,209.0
Loci shared by 5 taxa:  1,099.0
Loci shared by 6 taxa:  820.0
Loci shared by 7 taxa:  386.0
```

Again, we've got to make a decision here about how conservative we want to be regarding baits that hit all/some taxa. We only get 386 loci that we detect in all exemplars (including the base genome and the *menMol1* outgroup). That seems too strict (particularly because this total includes *menMol1*, which is really divergent from our taxa of interest). We also have to keep in mind that we can randomly fail to detect loci that are actually present, either by chance or do to sequence divergences that are >50% (the value we used in our search). In the end, I settled on loci we detected in $\geq$ 4 exemplar taxa. So we need to extract those from the database and store them in *triCas1+5-back-to-4.conf*:

```
phyluce_probe_query_multi_fasta_table \
    --db multifastas.sqlite \
    --base-taxon triCas1 \
    --output triCas1+5-back-to-4.conf \
    --specific-counts 4

Counter({'tricas1': 1160, 'anogla1': 1140, 'agrpla1': 1091, 'lepdec1': 1080, 'onttau1
↪': 1043, 'denpon1': 969, 'menmol1': 658})
Total loci = 1209
```

The values above show the number of loci detected in each exemplar taxon and the total number of loci we'll be targeting with the bait set we're about to design.

Your directory should look something like the following:

```
uce-coleoptera
+-- alignments (collapsed)
+-- bed  (collapsed)
+-- genomes  (collapsed)
+-- probe-design
    +-- coleoptera-genome.conf
    +-- coleoptera-genome-fasta
    |   +-- agrpla1.fasta
    |   +-- anogla1.fasta
    |   +-- denpon1.fasta
    |   +-- lepdec1.fasta
    |   +-- menmol1.fasta
    |   +-- onttau1.fasta
    |   +-- tricas1.fasta
    +-- coleoptera-genome-lastz
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_agrPla1.lastz.clean
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_anoGla1.lastz.clean
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_denPon1.lastz.clean
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_lepDec1.lastz.clean
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_menMol1.lastz.clean
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_ontTau1.lastz.clean
    |   +-- triCas1+5.temp-DUPE-SCREENED.probes_v_triCas1.lastz.clean
    +-- multifastas.sqlite
    +-- triCas1+5-back-to-4.conf
    +-- triCas1+5-back-to-4.conf.missing.matrix
    +-- triCas1+5+menMol1.sqlite
```

Final bait set design

Design a bait set using all exemplar genomes (and the base)

Now that we've settled on the set of loci we'll try to enrich, we want to design baits to target them. In contrast to the steps we took before to design the temporary bait set, we're using all of the exemplar genomes and the base genome to design probes. This way, we'll have a heterogeneous bait mix that contains probes designed from each exemplar but targeting the same locus, which should make the probe set we're designing more "universal".

To do this, we use a program similar to what we used before, except that this program has been modified to design probes across many exemplar genomes (instead of just one). As input, we give the program the name of the directory holding all of our fastas and the name of the config file we created in the step above. Then, as before, we need to add some metadata that will be incorporated to the bait set design file, and we tell the program to tile at 3x density, use a "middle" overlap, remove baits with >25% masking, and to design two probes targeting each locus. Finally, we write this probe set to a file named *coleoptera-v1-master-probe-list.fasta*.

```
phyluce_probe_get_tiled_probe_from_multiple_inputs \
    --fastas coleoptera-genome-fasta \
    --multi-fasta-output triCas1+5-back-to-4.conf \
    --probe-prefix "uce-" \
    --designer faircloth \
    --design coleoptera-v1 \
    --tiling-density 3 \
    --overlap middle \
    --masking 0.25 \
    --remove-gc \
    --two-probes \
    --output coleoptera-v1-master-probe-list.fasta

GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGNNGGGGGGGGGGGGGGGNGGGGGGGGGGGGNNGGGGGGGGGGGGGGGGGGGGGGGGGGGG

Conserved locus count = 1209
Probe Count = 14113
```

Note that the number of baits that we've designed to target 1209 conserved loci is quite high - this is because we're including roughly 2 baits for 1209 loci across 7 exemplar taxa (16926 is the theoretical maximum).

Remove duplicates from our bait set

As before, we need to check out bait set for duplicate loci. This time, the search is going to take longer, because of the larger number of baits. We'll align all the probes to themselves, then read in the alignments, and filter the probe list to remove putative duplicates.

Align probes to themselves at low stringency to identify duplicates:

```
phyluce_probe_easy_lastz \
    --target coleoptera-v1-master-probe-list.fasta \
    --query coleoptera-v1-master-probe-list.fasta \
    --identity 50 \
    --coverage 50 \
    --output coleoptera-v1-master-probe-list-TO-SELF-PROBES.lastz

Started:  Fri Jun 03, 2016  15:50:52
Ended:  Fri Jun 03, 2016  15:51:11
Time for execution:  0.322272149722 minutes
```

Now, screen the alignements and filter our master probe list to remove duplicates:

```
phyluce_probe_remove_duplicate_hits_from_probes_using_lastz \
    --fasta coleoptera-v1-master-probe-list.fasta \
    --lastz coleoptera-v1-master-probe-list-TO-SELF-PROBES.lastz \
    --probe-prefix=uce-

Parsing lastz file...
Screening results...
Screened 14112 fasta sequences.  Filtered 37 duplicates. Kept 13674.
```

The master probe list that has been filtered of putatively duplicate loci is now located in *coleoptera-v1-master-probe-list-DUPE-SCREENED.fasta*.

Your directory should look something like the following:

```
uce-coleoptera
+—— alignments (collapsed)
+—— bed  (collapsed)
+—— genomes  (collapsed)
+—— probe-design
    +—— coleoptera-genome.conf
    +—— coleoptera-genome-fasta
    │   +—— agrpla1.fasta
    │   +—— anogla1.fasta
    │   +—— denpon1.fasta
    │   +—— lepdec1.fasta
    │   +—— menmol1.fasta
    │   +—— onttau1.fasta
    │   +—— tricas1.fasta
    +—— coleoptera-genome-lastz
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_agrPla1.lastz.clean
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_anoGla1.lastz.clean
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_denPon1.lastz.clean
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_lepDec1.lastz.clean
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_menMol1.lastz.clean
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_ontTau1.lastz.clean
    │   +—— triCas1+5.temp-DUPE-SCREENED.probes_v_triCas1.lastz.clean
    +—— coleoptera-v1-master-probe-list-DUPE-SCREENED.fasta
    +—— coleoptera-v1-master-probe-list.fasta
    +—— coleoptera-v1-master-probe-list-TO-SELF-PROBES.lastz
    +—— multifastas.sqlite
    +—— triCas1+5-back-to-4.conf
    +—— triCas1+5-back-to-4.conf.missing.matrix
    +—— triCas1+5+menMol1.sqlite
```

The master bait list

What we've created, above, is the master bait list that contains baits targeting the conserved locus we identified. Because we've designed probes from multiple exemplar taxa, the number of overall baits is high (as high as 14 baits targeting each conserved locus). This bait set is ready for synthesis and subsequent enrichment of these conserved loci shared among coleoptera.

Subsetting the master probe list

Sometimes we might not want to synthesize *all* of the baits for *all* of the loci. For instance, we might be enriching loci from species that are nested within the clade defined by ((‘Anoplophora glabripennis (Asian longhorned beetle)’:’Leptinotarsa decemlineata (Colorado potato beetle)’)’Dendroctonus ponderosae (mountain pine beetle)’), and because we're only working with these species, we might want to drop the baits targeting UCE loci in Agrilus planipennis (emerald ash borer), Tribolium castaneum (red flour beetle), Onthophagus taurus (taurus scarab), and Mengenilla moldrzyki (Strepsiptera). This is actually pretty easy to do - we just need to subset the baits to include those taxa that we *do* want. Given the example, above, we can run:

```
> phyluce_probe_get_subsets_of_tiled_probes \
    --probes coleoptera-v1-master-probe-list-DUPE-SCREENED.fasta \
    --taxa anogla1 lepdec1 denpon1 \
    --output coleoptera-v1-master-probe-list-DUPE-SCREENED-SUBSET-CLADE_1.fasta

All probes = 13674
--- Probes by taxon ---
```

(continues on next page)

(continued from previous page)

```
anogla1 2189
menmol1 1236
lepdec1 2083
denpon1 1867
onttau1 1970
agrpla1 2086
tricas1 2243
--- Post  filtering ---
Conserved locus count = 1169
Probe Count = 6139
```

In-silico test of the bait design

Now that we've designed our baits, it's always good to run a sanity check on the data - if we use the baits to collect data from a selection of available genomes (or other genetic data), can we reconstruct a phylogeny that is sane, given what we know about the specific taxa?

How we do that is outlined below. Many of the steps we've run before, so I'm not going to explain these quite as much as I have previously. Several other of the steps that we're going to run are also outlined in *Tutorial I: UCE Phylogenomics*.

First, we need to make a directory to hold our in-silico test results:

```
> cd uce-coleoptera
> mkdir probe-design-test
> cd probe-design-test
```

Now, our directory tree should look something like:

```
uce-coleoptera
+—— alignments (collapsed)
+—— bed (collapsed)
+—— genomes  (collapsed)
+—— probe-design (collapsed)
+—— probe-design-test
```

Align our bait set to the extant genome sequences

Here (and if you have them), you may want to include *all* the genomic data you have access to - particularly if you removed some taxa because they were closely related to other taxa and designing probes from these closely related groups was redundant. To run these alignments (you've seen this before):

```
phyluce_probe_run_multiple_lastzs_sqlite \
    --db triCas1+5+strepsiptera-test.sqlite \
    --output coleoptera-genome-lastz \
    --probefile ../probe-design/coleoptera-v1-master-probe-list-DUPE-SCREENED.fasta \
    --scaffoldlist agrPla1 anoGla1 denPon1 lepDec1 ontTau1 triCas1 menMol1 \
    --genome-base-path ../genomes \
    --identity 50 \
    --cores 16
```

Warning: Note that I am using 16 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Now, we need to extract fasta data for each of these loci. This is effectively the same as what we've done before, but notice the use of *–flank* in place of *–probe*. This tells the program that we want to extract larger chunks of sequence, in thie base 400 bp to the each side of a given locus (if possible):

```
> phyluce_probe_slice_sequence_from_genomes \
    --conf coleoptera-genome.conf \
    --lastz coleoptera-genome-lastz \
    --output coleoptera-genome-fasta \
    --flank 400 \
    --name-pattern "coleoptera-v1-master-probe-list-DUPE-SCREENED.fasta_v_{}.lastz.
↪clean"

2016-06-03 16:36:47,712 - Phyluce - INFO - =================== Starting Phyluce:␣
↪Slice Sequence ===================
2016-06-03 16:36:47,714 - Phyluce - INFO - ------------------- Working on menMol1␣
↪genome -------------------
2016-06-03 16:36:47,715 - Phyluce - INFO - Reading menMol1 genome
2016-06-03 16:36:57,447 - Phyluce - INFO - menMol1: 766 uces, 73 dupes, 693 non-dupes,
↪ 0 orient drop, 2 length drop, 691 written
2016-06-03 16:36:57,447 - Phyluce - INFO - ------------------- Working on agrPla1␣
↪genome -------------------
2016-06-03 16:36:57,448 - Phyluce - INFO - Reading agrPla1 genome
2016-06-03 16:37:12,538 - Phyluce - INFO - agrPla1: 1151 uces, 81 dupes, 1070 non-
↪dupes, 0 orient drop, 34 length drop, 1036 written
2016-06-03 16:37:12,538 - Phyluce - INFO - ------------------- Working on anoGla1␣
↪genome -------------------
2016-06-03 16:37:12,539 - Phyluce - INFO - Reading anoGla1 genome
2016-06-03 16:37:29,320 - Phyluce - INFO - anoGla1: 1167 uces, 103 dupes, 1064 non-
↪dupes, 0 orient drop, 17 length drop, 1047 written
2016-06-03 16:37:29,321 - Phyluce - INFO - ------------------- Working on denPon1␣
↪genome -------------------
2016-06-03 16:37:29,322 - Phyluce - INFO - Reading denPon1 genome
2016-06-03 16:37:44,958 - Phyluce - INFO - denPon1: 1126 uces, 174 dupes, 952 non-
↪dupes, 1 orient drop, 16 length drop, 935 written
2016-06-03 16:37:44,959 - Phyluce - INFO - ------------------- Working on lepDec1␣
↪genome -------------------
2016-06-03 16:37:44,959 - Phyluce - INFO - Reading lepDec1 genome
2016-06-03 16:38:01,794 - Phyluce - INFO - lepDec1: 1156 uces, 142 dupes, 1014 non-
↪dupes, 6 orient drop, 22 length drop, 986 written
2016-06-03 16:38:01,794 - Phyluce - INFO - ------------------- Working on ontTau1␣
↪genome -------------------
2016-06-03 16:38:01,796 - Phyluce - INFO - Reading ontTau1 genome
2016-06-03 16:38:16,611 - Phyluce - INFO - ontTau1: 1134 uces, 100 dupes, 1034 non-
↪dupes, 3 orient drop, 14 length drop, 1017 written
2016-06-03 16:38:16,612 - Phyluce - INFO - ------------------- Working on triCas1␣
↪genome -------------------
2016-06-03 16:38:16,613 - Phyluce - INFO - Reading triCas1 genome
2016-06-03 16:38:32,786 - Phyluce - INFO - triCas1: 1172 uces, 70 dupes, 1102 non-
↪dupes, 13 orient drop, 13 length drop, 1076 written
```

Match contigs to baits

In the step above, we essentially extracted FASTA data for each taxon, and wrote those out into individual FASTA files. These are the equivalent of the assembled contigs that we use in the standard phyluce pipeline, so now, we're going to use that workflow. Note that the filtering in *phyluce_assembly_match_contigs_to_probes* is more strict that what we used above to identify contigs.

```
> phyluce_assembly_match_contigs_to_probes \
    --contigs coleoptera-genome-fasta \
    --probes ../probe-design/coleoptera-v1-master-probe-list-DUPE-SCREENED.fasta \
    --output in-silico-lastz \
    --min_coverage 67 \
    --log-path log

2016-06-03 16:40:36,888 - phyluce_assembly_match_contigs_to_probes - INFO - agrpla1:␣
↪903 (87.16%) uniques of 1036 contigs, 0 dupe probe matches, 116 UCE loci removed␣
↪for matching multiple contigs, 117 contigs removed for matching multiple UCE loci
2016-06-03 16:41:06,688 - phyluce_assembly_match_contigs_to_probes - INFO - anogla1:␣
↪927 (88.54%) uniques of 1047 contigs, 0 dupe probe matches, 111 UCE loci removed␣
↪for matching multiple contigs, 116 contigs removed for matching multiple UCE loci
2016-06-03 16:41:28,524 - phyluce_assembly_match_contigs_to_probes - INFO - denpon1:␣
↪819 (87.59%) uniques of 935 contigs, 0 dupe probe matches, 85 UCE loci removed for␣
↪matching multiple contigs, 89 contigs removed for matching multiple UCE loci
2016-06-03 16:41:54,879 - phyluce_assembly_match_contigs_to_probes - INFO - lepdec1:␣
↪900 (91.28%) uniques of 986 contigs, 0 dupe probe matches, 80 UCE loci removed for␣
↪matching multiple contigs, 81 contigs removed for matching multiple UCE loci
2016-06-03 16:42:05,300 - phyluce_assembly_match_contigs_to_probes - INFO - menmol1:␣
↪527 (76.27%) uniques of 691 contigs, 0 dupe probe matches, 58 UCE loci removed for␣
↪matching multiple contigs, 58 contigs removed for matching multiple UCE loci
2016-06-03 16:42:29,353 - phyluce_assembly_match_contigs_to_probes - INFO - onttau1:␣
↪854 (83.97%) uniques of 1017 contigs, 0 dupe probe matches, 127 UCE loci removed␣
↪for matching multiple contigs, 130 contigs removed for matching multiple UCE loci
2016-06-03 16:43:01,303 - phyluce_assembly_match_contigs_to_probes - INFO - tricas1:␣
↪934 (86.80%) uniques of 1076 contigs, 0 dupe probe matches, 140 UCE loci removed␣
↪for matching multiple contigs, 141 contigs removed for matching multiple UCE loci
```

Get match counts and extract FASTA information

Now, we need to get the count of matches that we recovered to UCE loci in the probe set, and extract all of the "good" loci to a monolithic FASTA (see *Tutorial I: UCE Phylogenomics* if this is not making sense):

```
phyluce_assembly_get_match_counts \
    --locus-db in-silico-lastz/probe.matches.sqlite \
    --taxon-list-config in-silico-coleoptera-taxon-sets.conf \
    --taxon-group 'all' \
    --output taxon-sets/insilico-incomplete/insilico-incomplete.conf \
    --log-path log \
    --incomplete-matrix

2016-06-03 16:50:02,610 - phyluce_assembly_get_match_counts - INFO - There are 7 taxa␣
↪in the taxon-group '[all]' in the config file in-silico-coleoptera-taxon-sets.conf
2016-06-03 16:50:02,610 - phyluce_assembly_get_match_counts - INFO - Getting UCE␣
↪names from database
2016-06-03 16:50:02,617 - phyluce_assembly_get_match_counts - INFO - There are 1172␣
↪total UCE loci in the database
```

(continues on next page)

(continued from previous page)

```
2016-06-03 16:50:02,708 - phyluce_assembly_get_match_counts - INFO - Getting UCE␣
↪matches by organism to generate a INCOMPLETE matrix
2016-06-03 16:50:02,709 - phyluce_assembly_get_match_counts - INFO - There are 1093␣
↪UCE loci in an INCOMPLETE matrix
2016-06-03 16:50:02,709 - phyluce_assembly_get_match_counts - INFO - Writing the taxa␣
↪and loci in the data matrix to /nfs/data1/working/bfaircloth-insects/coleoptera/
↪triCas1+5+strepsiptera-test/taxon-sets/insilico-incomplete/insilico-incomplete.conf
```

Now, extract the FASTA information for each locus into a monolithic FASTA file:

```
phyluce_assembly_get_fastas_from_match_counts \
    --contigs ../../coleoptera-genome-fasta \
    --locus-db ../../in-silico-lastz/probe.matches.sqlite \
    --match-count-output insilico-incomplete.conf \
    --output insilico-incomplete.fasta \
    --incomplete-matrix insilico-incomplete.incomplete \
    --log-path log
```

Align the conserved locus data

Now, we need to align the sequence data for each conserved locus in our data set. We'll do this using standard phyluce tools (mafft). First, change into the working directory:

```
cd taxon-sets/insilico-incomplete
```

Now, align the sequences:

```
phyluce_align_seqcap_align \
    --fasta insilico-incomplete.fasta \
    --output mafft \
    --taxa 7 \
    --incomplete-matrix \
    --cores 12 \
    --no-trim \
    --output-format fasta \
    --log-path log
```

Warning: Note that I am using 12 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Trim the conserved locus alignments

Still following the standard phyluce workflow, trim the resulting alignments:

```
phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed \
    --alignments mafft \
    --output mafft-gblocks \
    --b1 0.5 \
    --b4 8 \
    --cores 12 \
    --log log
```

Warning: Note that I am using 12 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Remove the locus names from each alignment

And, remove the locus names from each of the resulting alignments:

```
phyluce_align_remove_locus_name_from_files \
    --alignments mafft-gblocks \
    --output mafft-gblocks-clean \
    --cores 12 \
    --log-path log
```

Warning: Note that I am using 12 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Get stats across the aligned loci

Compute stats across the alignments:

```
python ~/git/phyluce/bin/align/phyluce_align_get_align_summary_data \
    --alignments mafft-gblocks-clean \
    --cores 12 \
    --log-path log

2016-06-03 16:57:11,675 - phyluce_align_get_align_summary_data - INFO - =========␣
↪Starting phyluce_align_get_align_summary_data =========
2016-06-03 16:57:11,675 - phyluce_align_get_align_summary_data - INFO - Version: git␣
↪6ab3a4b
2016-06-03 16:57:11,675 - phyluce_align_get_align_summary_data - INFO - Argument --
↪alignments: triCas1+5+strepsiptera-test/taxon-sets/insilico-incomplete/mafft-
↪gblocks-clean
2016-06-03 16:57:11,675 - phyluce_align_get_align_summary_data - INFO - Argument --
↪cores: 12
2016-06-03 16:57:11,675 - phyluce_align_get_align_summary_data - INFO - Argument --
↪input_format: nexus
2016-06-03 16:57:11,675 - phyluce_align_get_align_summary_data - INFO - Argument --
↪log_path: triCas1+5+strepsiptera-test/taxon-sets/insilico-incomplete/log
2016-06-03 16:57:11,676 - phyluce_align_get_align_summary_data - INFO - Argument --
↪output: None
2016-06-03 16:57:11,676 - phyluce_align_get_align_summary_data - INFO - Argument --
↪show_taxon_counts: False
2016-06-03 16:57:11,676 - phyluce_align_get_align_summary_data - INFO - Argument --
↪verbosity: INFO
2016-06-03 16:57:11,676 - phyluce_align_get_align_summary_data - INFO - Getting␣
↪alignment files
2016-06-03 16:57:11,710 - phyluce_align_get_align_summary_data - INFO - Computing␣
↪summary statistics using 12 cores
2016-06-03 16:57:15,381 - phyluce_align_get_align_summary_data - INFO - -------------
↪--------- Alignment summary ----------------------
2016-06-03 16:57:15,382 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪loci:      994
```

(continues on next page)

(continued from previous page)

```
2016-06-03 16:57:15,382 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪length:    644,447
2016-06-03 16:57:15,382 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪mean:      648.34
2016-06-03 16:57:15,383 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪95% CI:    9.39
2016-06-03 16:57:15,383 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪min:       240
2016-06-03 16:57:15,383 - phyluce_align_get_align_summary_data - INFO - [Alignments]␣
↪max:       1,444
2016-06-03 16:57:15,383 - phyluce_align_get_align_summary_data - INFO - --------------
↪----- Informative Sites summary --------------------
2016-06-03 16:57:15,384 - phyluce_align_get_align_summary_data - INFO - [Sites] loci:␣
↪  994
2016-06-03 16:57:15,384 - phyluce_align_get_align_summary_data - INFO - [Sites]␣
↪total:  169,024
2016-06-03 16:57:15,384 - phyluce_align_get_align_summary_data - INFO - [Sites] mean:␣
↪  170.04
2016-06-03 16:57:15,384 - phyluce_align_get_align_summary_data - INFO - [Sites] 95%␣
↪CI: 4.48
2016-06-03 16:57:15,384 - phyluce_align_get_align_summary_data - INFO - [Sites] min: ␣
↪  0
2016-06-03 16:57:15,384 - phyluce_align_get_align_summary_data - INFO - [Sites] max: ␣
↪  390
2016-06-03 16:57:15,386 - phyluce_align_get_align_summary_data - INFO - --------------
↪----------- Taxon summary -------------------------
2016-06-03 16:57:15,386 - phyluce_align_get_align_summary_data - INFO - [Taxa] mean: ␣
↪          5.76
2016-06-03 16:57:15,386 - phyluce_align_get_align_summary_data - INFO - [Taxa] 95%␣
↪CI:          0.07
2016-06-03 16:57:15,386 - phyluce_align_get_align_summary_data - INFO - [Taxa] min:  ␣
↪          3
2016-06-03 16:57:15,386 - phyluce_align_get_align_summary_data - INFO - [Taxa] max:  ␣
↪          7
2016-06-03 16:57:15,387 - phyluce_align_get_align_summary_data - INFO - --------------
↪--- Missing data from trim summary ----------------
2016-06-03 16:57:15,387 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪mean: 0.00
2016-06-03 16:57:15,387 - phyluce_align_get_align_summary_data - INFO - [Missing] 95%␣
↪CI:       0.00
2016-06-03 16:57:15,387 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪min:  0.00
2016-06-03 16:57:15,387 - phyluce_align_get_align_summary_data - INFO - [Missing]␣
↪max:  0.00
2016-06-03 16:57:15,399 - phyluce_align_get_align_summary_data - INFO - --------------
↪------ Character count summary --------------------
2016-06-03 16:57:15,399 - phyluce_align_get_align_summary_data - INFO - [All␣
↪characters]       3,655,040
2016-06-03 16:57:15,399 - phyluce_align_get_align_summary_data - INFO - [Nucleotides]␣
↪          3,518,743
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - --------------
↪-- Data matrix completeness summary ---------------
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 50%] ␣
↪          946 alignments
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 55%] ␣
↪          946 alignments
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 60%] ␣
↪          865 alignments
```

(continues on next page)

(continued from previous page)

```
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 65%] ␣
↪           865 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 70%] ␣
↪           865 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 75%] ␣
↪           657 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 80%] ␣
↪           657 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 85%] ␣
↪           657 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 90%] ␣
↪           275 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 95%] ␣
↪           275 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - --------------
↪---------- Character counts -----------------------
2016-06-03 16:57:15,399 - phyluce_align_get_align_summary_data - INFO - [All␣
↪characters]        3,655,040
2016-06-03 16:57:15,399 - phyluce_align_get_align_summary_data - INFO - [Nucleotides]␣
↪           3,518,743
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - --------------
↪-- Data matrix completeness summary ---------------
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 50%] ␣
↪           946 alignments
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 55%] ␣
↪           946 alignments
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 60%] ␣
↪           865 alignments
2016-06-03 16:57:15,400 - phyluce_align_get_align_summary_data - INFO - [Matrix 65%] ␣
↪           865 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 70%] ␣
↪           865 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 75%] ␣
↪           657 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 80%] ␣
↪           657 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 85%] ␣
↪           657 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 90%] ␣
↪           275 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Matrix 95%] ␣
↪           275 alignments
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - --------------
↪---------- Character counts -----------------------
2016-06-03 16:57:15,401 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'-' is present 136,297 times
2016-06-03 16:57:15,402 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'A' is present 1,047,965 times
2016-06-03 16:57:15,402 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'C' is present 708,469 times
2016-06-03 16:57:15,402 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'G' is present 706,567 times
2016-06-03 16:57:15,402 - phyluce_align_get_align_summary_data - INFO - [Characters]
↪'T' is present 1,055,742 times
2016-06-03 16:57:15,402 - phyluce_align_get_align_summary_data - INFO - =========␣
↪Completed phyluce_align_get_align_summary_data ========
```

Warning: Note that I am using 12 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Generate an incomplete matrix

Now, given the alignments that we have, let's generate a 70% complete matrix:

```
phyluce_align_get_only_loci_with_min_taxa \
    --alignments mafft-gblocks-clean \
    --taxa 7 \
    --output mafft-gblocks-70p \
    --percent 0.75 \
    --cores 12 \
    --log log

2016-06-03 16:58:19,687 - phyluce_align_get_only_loci_with_min_taxa - INFO - Copied␣
↪865 alignments of 994 total containing ≥ 0.75 proportion of taxa (n = 5)
```

Warning: Note that I am using 12 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Prep raxml files, run raxml ML searches, and reconcile best tree w/ bootreps

Setup the PHYLIP-formatted files for raxml:

```
phyluce_align_concatenate_alignments \
    --alignments mafft-gblocks-70p \
    --output mafft-gblocks-70p-raxml \
    --log-path log --phylip
```

Now, run raxml against this phylip file

```
raxmlHPC-PTHREADS-SSE3 -m GTRGAMMA -N 20 -p 772374015 -n BEST -s mafft-gblocks-70p.
↪phylip -o menmol1 -T 10
raxml/raxmlHPC-PTHREADS-SSE3 -m GTRGAMMA -N autoMRE -p 772374015 -b 444353738 -n␣
↪bootrep -s mafft-gblocks-70p.phylip -o menmol1 -T 10
```

Warning: Note that I am using 10 physical CPU cores (*–cores*) to do this work. You need to use the number of physical cores available on *your* machine.

Now, reconcile the best ML tree w/ the bootreps:

```
raxmlHPC-SSE3 -f b \
    -m GTRGAMMA \
    -t RAxML_bestTree.BEST \
    -z RAxML_bootstrap.bootrep \
    -n FINAL -o menmol1
```

And rename the tips. To do this, setup a config file with the old and new names, like:

```
[all]
agrpla1:Agrilus planipennis (emerald ash borer)
anogla1:Anoplophora glabripennis (Asian longhorned beetle)
denpon1:Dendroctonus ponderosae (mountain pine beetle)
lepdec1:Leptinotarsa decemlineata (Colorado potato beetle)
menmol1:Mengenilla moldrzyki (Strepsiptera)
onttau1:Onthophagus taurus (taurus scarab)
tricas1:Tribolium castaneum (red flour beetle)
```

And rename the tips:

```
phyluce_genetrees_rename_tree_leaves \
    --order left:right \
    --input-format newick \
    --output-format newick \
    --config rename.conf \
    --section all \
    --input RAxML_bipartitions.FINAL \
    --output RAxML_bipartitions.NAME.FINAL.tre
```

3.4 Phyluce in Daily Use

3.4.1 Quality Control

When you receive your data from the sequencer, typically they are already demultiplexed (split by sequence tags) for you. If your data are from the MiSeq, they may also have been trimmed of adapter contamination.

Note: If your data are not demultiplexed, they can come in a vast array of different types, although one common type are so-called BCL-files. Regardless, if your data are not demultiplexed fastq files, you will need to talk to your sequencing provider about how to accomplish demultiplexing. I provide an **unsupported** guide to demultiplexing *BCL* files using Casava or bcl2fastq here and a guide to demultiplexing fastq data here

Regardless, you need to do a fair bit of quality control on your read data. **At a minimum**, this includes:

- getting some idea of how much data you have
- trimming adapter contamination of reads
- trimming low quality bases from reads

Although the MiSeq may trim some adapter contamination, running your reads through an additional round of trimming won't hurt. There is also lots of evidence showing that quality control of your read data has a **large** effect on your overall success, particularly for the most common way of working with data in phyluce. Bottom line is: *garbage in, garbage out*.

Read Counts

The first thing to do once you have your read data in hand is to to get an idea of the split of reads among your samples. This does two things:

1. Allows you to determine how well you split the run among your indexes/sequence tags
2. Shows you which samples may be suboptimal (have very few reads)

Really unequal read counts mean that you may want to switch up your library quantification process (or your pooling steps, prior to enrichment). Suboptimal read counts for particular libraries may or may not mean that the enrichments of those samples "failed", but it's generally a reasonable indication.

You can get read counts in one of two way - the first is very simple, the second uses phyluce.

Count reads using shell tools

You can get a quick and dirty idea of the number of reads you have for each sample using simple shell or "terminal" tools - counting lines in lots of files is a task that's really suited to UNIX-like operating systems. As mentioned in *Tutorial I: UCE Phylogenomics* section, we can do this several ways. We'll use tools from unix, because they are fast. The next line of code will count the lines in each R1 file (which should be equal to the reads in the R2 file) and divide that number by 4 to get the number of sequence reads.

```
for i in *_R1_*.fastq.gz; do echo $i; gunzip -c $i | wc -l | awk '{print $1/4}'; done
```

The number of reads in the R2 files, if you have paired-end data, should **always** be equal.

Get read counts using phyluce

There is a bit of code in phyluce that lets you count reads. To use it, you pass the code the directory containing reads you want to summarize and run:

```
phyluce_assembly_get_fastq_lengths --input /directory/containing/reads/ --csv; done
```

You can run this across many directories of reads as described in *Tutorial I: UCE Phylogenomics*.

Adapter- and quality-trimming

Generally speaking, Casava and bcl2fastq, the two programs used to demultiplex sequence data from Illumina platforms, output fastq files in a format similar to the following:

```
Project_name/
    Sample_BFIDT-000
        BFIDT-000_AGTATAGCGC_L001_R1_001.fastq.gz
        BFIDT-000_AGTATAGCGC_L001_R2_001.fastq.gz
    Sample_BFIDT-001
        BFIDT-001_TTGTTGGCGG_L001_R1_001.fastq.gz
        BFIDT-001_TTGTTGGCGG_L001_R2_001.fastq.gz
```

What you want to do is to clean these reads of adapter contamination and trim low-quality bases from all reads (and probably also drop reads containing "N" (ambiguous) bases. Then you want to interleave the resulting data, where read pairs are maintained, and also have an output file of singleton data, where read pairs are not.

You can do this however you like. phyluce assumes that the results structure of your data after trimming will look like the following (replace genus_species with your taxon names):

```
genus_species1/
    split-adapter-quality-trimmed/
        genus_species1-READ1.fastq.gz
        genus_species1-READ2.fastq.gz
        genus_species1-READ-singleton.fastq.gz
genus_species2/
    split-adapter-quality-trimmed/
```

(continues on next page)

(continued from previous page)

```
        genus_species2-READ1.fastq.gz
        genus_species2-READ2.fastq.gz
        genus_species2-READ-singleton.fastq.gz
```

This can be accomplished in an automated fashion using illumiprocessor.

Trimming with illumiprocessor

You can run your adapter and quality trimming and output the files in the correct format using a program I wrote called illumiprocessor. It automates these processes over hundred of files and produces output in the format we want downstream.

You need to generate a configuration file `your-illumiprocessor.conf`, that gives details of your reads, how you want them processed, and what renaming options to use. There are **several** variations in formatting required depending on the library preparation method that you used.

Attention: See the documentation for illumiprocessor for configuration information.

You can run illumiprocessor against your data (in *demultiplexed*) with the following. If you do not have a multicore machine, you may want to run with `--cores=1`. Additionally, multicore operations require a fair amount of RAM, so if you're low on RAM, run with fewer cores:

```
illumiprocessor \
    --input demultiplexed \
    --output uce-clean \
    --config your-illumiprocesser.conf \
    --cores 12
```

The clean data will appear in `uce-clean` with the following structure:

```
uce-clean/
    genus_species1/
        adapters.fasta
        raw-reads/
            genus_species1-READ1.fastq.gz (symlink)
            genus_species1-READ2.fastq.gz (symlink)
        split-adapter-quality-trimmed/
            genus_species1-READ1.fastq.gz
            genus_species1-READ2.fastq.gz
            genus_species1-READ-singleton.fastq.gz
        stats/
            genus_species1-adapter-contam.txt

    genus_species2/
        adapters.fasta
        raw-reads/
            genus_species2-READ1.fastq.gz (symlink)
            genus_species2-READ2.fastq.gz (symlink)
        split-adapter-quality-trimmed/
            genus_species2-READ1.fastq.gz
            genus_species2-READ2.fastq.gz
            genus_species2-READ-singleton.fastq.gz
```

(continues on next page)

(continued from previous page)

```
        stats/
            genus_species2-adapter-contam.txt
```

You are now ready to move onto *Assembly* of the cleaned read data.

3.4.2 Assembly

Setup

Once your reads are clean, you're ready to assemble. At the moment, you can use velvet, ABySS, and spades for assembly.

Most of the assembly process is automated using code within phyluce, specifically the following 3 scripts:

- *phyluce_assembly_assemblo_abyss*
- *phyluce_assembly_assemblo_spades*
- *phyluce_assembly_assemblo_velvet*

The code of each of the above programs **always** expects your input directories to have the following structure (from the *Quality Control* section):

```
uce-clean/
    genus_species1/
        adapters.fasta
        raw-reads/
            genus_species1-READ1.fastq.gz (symlink)
            genus_species1-READ2.fastq.gz (symlink)
        split-adapter-quality-trimmed/
            genus_species1-READ1.fastq.gz
            genus_species1-READ2.fastq.gz
            genus_species1-READ-singleton.fastq.gz
        stats/
            genus_species1-adapter-contam.txt
```

And, each of these assembly helper programs take the same configuration files as input. You should format the configuration file for input according to the following scheme:

```
[samples]
name_you_want_assembly_to_have:/path/to/uce-clean/genus_species1
```

In practice, this means you need to create a configuration file that looks like:

```
[samples]
anas_platyrhynchos1:/path/to/uce-clean/anas_platyrhynchos1
anas_carolinensis1:/path/to/uce-clean/anas_carolinensis1
dendrocygna_bicolor1:/path/to/uce-clean/dendrocygna_bicolor1
```

The *assembly name* on the left side of the colon can be whatever you want. The *path name* on the right hand side of the colon must be a valid path to a directory containing read data in a format similar to that described above.

Attention: Assembly names **MUST** be unique.

Question: How do I name my samples/assemblies?

Naming samples is a contentious issue and is also a hard thing to deal with using computer code. You should **never** have a problem if you name your samples as follows, where the genus and specific epithet are separated by an underscore, and multiple individuals of a given species are indicated using a trailing integer value:

```
anas_platyrhynchos1
anas_carolinensis1
dendrocygna_bicolor1
```

You should also not have problems if you use a naming scheme that suffixes the species binomial(s) with an accession number that is **simply** formatted (e.g. no slashes, dashes, etc.):

```
anas_platyrhynchos_KGH2267
anas_carolinensis_KGH2269
dendrocygna_bicolor_DWF4597
```

The above is the recommended working format. When you search for UCE contigs phyluce should screen your taxon name to ensure they do not contain restricted characters. This includes `.+:"'-?!*@%^&#=/\` or names that **begin** with a number. It's probably best to get that all squared away now.

Running the assembly

Once your configuration file is created (best to use a decent text editor that will not cause you grief), you are ready to start assembling your read data into contigs that we will search for UCEs. The code to do this for the three helper scripts is below.

General process

The general process that the helper scripts use is:

1. Create the output directory (AKA $ASSEMBLY, below)
2. Create a `contigs` folder within the output directory
3. For each taxon create `$ASSEMBLY/genus-species` directory, based on config file entries
4. Find the correct fastq files for a given sample
5. Input those fastq files to whichever assembly program
6. Assemble reads
7. Strip contigs of potentially problematic bases (ABySS-only)
8. Normalize contig names
9. Link all assembly files with normalized names in $ASSEMBLY/genus-species/ into $ASSEMBLY/contigs/genus-species.contigs.fasta, so that all assemblies are linked in the same output directory.

velvet

```
# make a directory for log files
mkdir log
# run the assembly
phyluce_assembly_assemblo_velvet \
    --config config_file_you_created.conf \
    --output /path/where/you/want/assemblies \
    --kmer 35 \
    --subfolder split-adapter-quality-trimmed \
    --cores 12 \
    --clean \
    --log-path log
```

Results

The directory structure created for velvet-based assemblies looks like:

```
path-to-output-directory/
    contigs/
        genus-species1 -> ../genus-species1/out_k31/contigs.fa
    genus-species1/
        contigs.fasta -> out_k31/contigs.fa
        out_k31
        velvetg-k31.err.log
        velvetg-k31.out.log
        velveth-k31.err.log
        velveth-k31.out.log
```

ABySS

```
# make a directory for log files
mkdir log
# run the assembly
phyluce_assembly_assemblo_abyss \
    --config config_file_you_created.conf \
    --output /path/where/you/want/assemblies \
    --kmer 35 \
    --subfolder split-adapter-quality-trimmed \
    --cores 12 \
    --clean \
    --log-path log
```

Attention: Following assembly, *phyluce_assembly_assemblo_abyss* modifies the assemblies by replacing degenerate base codes with standard nucleotide encodings. We do this because lastz, which we use to match contigs to targeted UCE loci, is not compatible with degenerate IUPAC codes.

The phyluce_assembly_assemblo_abyss code makes these substitutions for every site having a degenerate code by selecting the appropriate nucleotide encoding randomly. The code also renames the ABySS assemblies using the velvet naming convention. The modified contigs are them symlinked into `$ASSEMBLY/contigs`. Unmodified contigs are available in `$ASSEMBLY/genus- species/out_k*-contigs.fa`

Results

The directory structure created for ABySS-based assemblies looks like:

```
path-to-output-directory/
    contigs/
        genus-species1 -> ../genus-species1/out_k31-contigs-velvet.fa
    genus-species1/
        abyss-k31.err.log
        contigs.fasta -> out_k31-contigs-velvet.fa
        out_k31-contigs.fa
        out_k31-scaffolds.fa
        out_k31-unitigs.fa
        abyss-k31.out.log
        coverage.hist
        out_k31-contigs-velvet.fa
        out_k31-stats
```

Spades

```
# make a directory for log files
mkdir log
# run the assembly
phyluce_assembly_assemblo_spades \
    --config config_file_you_created.conf \
    --output /path/where/you/want/assemblies \
    --subfolder split-adapter-quality-trimmed \
    --clean \
    --cores 12 \
    --log-path log
```

Results

The directory structure created for spades-based assemblies looks like:

```
path-to-output-directory/
    contigs/
        genus-species1 -> ../genus-species1/scaffolds.fasta
    genus-species1/
        contigs.fasta -> Trinity.fasta
        Trinity.fasta
        trinity.log
```

Common questions

Question: Which assembly program do I pick?

Generally, I would suggest that you use spades. It produces reasonable contig assemblies that are longer than the assemblies built by velvet, ABySS, or Trinity (now removed from phyluce). It arguable produces assemblies that are more accurate than assemblies from these other programs.

Question: For ABySS and velvet, what –kmer value do I use?

Also a hard question. Part of the reason that it is hard is due to the fact that we are trying to assemble data of heterogenous read depth (i.e., our reads are spread across (mostly) UCE loci, but the depth of coverage of each locus is varaible due to capture efficiency). Longer kmer values can give you longer (but fewer) contigs, while shorter kmer values produce fewer, more abundant contigs. In most cases, your assemblies will be decent with a kmer value around 55-65.

3.4.3 UCE Processing for Phylogenomics

The process described below is meant for users who are analyzing UCE data in phylogenetic contexts - meaning that you are interested in addressing questions at or deeper than the species-level.

Identifying UCE loci

Once we have assembled our fastq data (see *Assembly*), we need to process those contigs to (a) determine which represent enrichend UCE loci and (b) remove any potential paralogs from the data set. Before we can do that, we need to to a little preparatory work by downloading a FASTA file representing the bait/probe set that we used.

Get the probe set FASTA

To identify which of the contigs we've assembled are UCE loci (and which UCE loci they might be), we are going to match our assembled contigs to the probes we used to enrich UCE loci. Before we do that, however, we need to download a copy of probe set we used for matching purposes.

Attention: We archive official probe sets at https://github.com/faircloth-lab/uce-probe-sets/, but you need to be careful about which one you grab - probe sets can be of different sizes (e.g. 2,500 or 5,500 loci) and for different groups of taxa (e.g., amniotes, fish)

Download the probe set

To download a given probe set for phyluce, you need to figure out which probe set you need. Then, you can use a command like `wget` on the command-line (or navigate with your browser to the URL and save the file):

```
# to get the 2.5k, amniote probe set
wget https://raw.githubusercontent.com/faircloth-lab/uce-probe-sets/master/uce-2.5k-
↪probe-set/uce-2.5k-probes.fasta

# to get the 5k, amniote probe set
wget https://raw.githubusercontent.com/faircloth-lab/uce-probe-sets/master/uce-5k-
↪probe-set/uce-5k-probes.fasta
```

Match contigs to probes

Once we've downloaded the probe set we used to enrich UCE loci, we need to find which of our assembled contigs are the UCE loci that we enriched. During this process, the code will also remove any contigs that appear to be duplicates as a result of assembly/other problems **or** a biological event(s).

The way that this process works is that phyluce aligns (using lastz) the contigs you assembled to the probes you input on a taxon-by-taxon (or otu-by-otu)

> basis. Then, the code parses the alignment file to determine which contigs

matched which probes, whether any probes from a single locus matched multiple contigs or whether a single contig matched probes designed from muliple UCE loci. Either of these latter two events suggests that the locus in question is problematic.

Hint: ADVANCED: The default regular expression assumes probes in your file are named according to `uce-NNN_pN`, where `uce-` is just a text string, `NNN` is an integer value denoting each unique locus, `_p` is a text string denoting a "probe" targeting locus `NNN`, and the trailing `N` is an integer value denoting each unique probe targeting the same locus.

If you are using a custom probe file, then you will either need to ensure that your naming scheme conforms to this approach **OR** you will need to input a different regular expression to convert the probe names to locus names using the `--regex` flag. **It is up to you to determine what is the appropriate regular expression**.

To identify which of your assembled contigs are UCE contigs, run:

```
# make a directory for log files
mkdir log
# match contigs to probes
phyluce_assembly_match_contigs_to_probes \
    --contigs /path/to/assembly/contigs/ \
    --probes uce-5k-probes.fasta \
    --output /path/to/uce/output \
    --log-path log
```

When you run this code, you should see output similar to:

```
2014-04-24 14:38:15,979 - match_contigs_to_probes - INFO - ================ Starting␣
↪match_contigs_to_probes ===============
2014-04-24 14:38:15,979 - match_contigs_to_probes - INFO - Version: git 7aec8f1
2014-04-24 14:38:15,979 - match_contigs_to_probes - INFO - Argument --contigs: /path/
↪to/assembly/contigs/
2014-04-24 14:38:15,980 - match_contigs_to_probes - INFO - Argument --keep_
↪duplicates: None
2014-04-24 14:38:15,980 - match_contigs_to_probes - INFO - Argument --log_path: None
2014-04-24 14:38:15,980 - match_contigs_to_probes - INFO - Argument --min_coverage: 80
2014-04-24 14:38:15,980 - match_contigs_to_probes - INFO - Argument --min_identity: 80
2014-04-24 14:38:15,980 - match_contigs_to_probes - INFO - Argument --output: /path/
↪to/uce/output
2014-04-24 14:38:15,980 - match_contigs_to_probes - INFO - Argument --probes: uce-5k-
↪probes.fasta
2014-04-24 14:38:15,981 - match_contigs_to_probes - INFO - Argument --regex: ^(uce-
↪\d+)(?:_p\d+.*)
2014-04-24 14:38:15,981 - match_contigs_to_probes - INFO - Argument --verbosity: INFO
2014-04-24 14:38:16,138 - match_contigs_to_probes - INFO - Checking probe/bait␣
↪sequences for duplicates
2014-04-24 14:38:19,022 - match_contigs_to_probes - INFO - Creating the UCE-match␣
↪database
2014-04-24 14:38:19,134 - match_contigs_to_probes - INFO - Processing contig data
2014-04-24 14:38:19,134 - match_contigs_to_probes - INFO - ---------------------------
↪--------------------------------------
2014-04-24 14:38:25,713 - match_contigs_to_probes - INFO - genus_species1: 1031 (70.14
↪%) uniques of 1470 contigs, 0 dupe probe matches, 48 UCE probes matching multiple␣
↪contigs, 117 contigs matching multiple UCE probes
```

(continues on next page)

(continued from previous page)

```
2014-04-24 14:38:32,846 - match_contigs_to_probes - INFO - genus_species2: 420 (68.52
↪%) uniques of 613 contigs, 0 dupe probe matches, 30 UCE probes matching multiple␣
↪contigs, 19 contigs matching multiple UCE probes
2014-04-24 14:38:39,184 - match_contigs_to_probes - INFO - genus_species3: 1071 (63.15
↪%) uniques of 1696 contigs, 0 dupe probe matches, 69 UCE probes matching multiple␣
↪contigs, 101 contigs matching multiple UCE probes
2014-04-24 14:49:59,654 - match_contigs_to_probes - INFO - --------------------------
↪--------------------------------------
2014-04-24 14:49:59,654 - match_contigs_to_probes - INFO - The LASTZ alignments are␣
↪in /path/to/uce/output/
2014-04-24 14:49:59,654 - match_contigs_to_probes - INFO - The UCE match database is␣
↪in /path/to/uce/output/probe.matches.sqlite
2014-04-24 14:49:59,655 - match_contigs_to_probes - INFO - =============== Completed␣
↪match_contigs_to_probes ===============
```

Note: The `*.log` files for each operation are always printed to the screen AND also written out to the `$CWD` (current working directory). You can keep these files more orderly by specifying a `$LOG` on the command line using the `--log-path` option.

Results

The resulting files will be in the:

```
/path/to/output
```

directory. If you look in this directory, you'll see that it contains species- specific *lastz_* files as well as an sqlite database:

```
$ ls /path/to/output

genus_species1.contigs.lastz
genus_species2.contigs.lastz
genus_species3.contigs.lastz
probe.matches.sqlite
```

The `*.lastz` files within the `/path/to/output` directory are basically for reference and individual review (they are text files that you can open using a text editor to view). The really important data from the lastz files are summarized in the:

```
probe.matches.sqlite
```

database file. It's probably a good idea to have some knowledge of how this database is structured, since it's basically what makes the next few steps work. So, let's go over the structure and contents of this database.

The probe.matches.sqlite database

`probe.matches.sqlite` is a relational database that summarizes all **valid** matches of contigs to UCE loci across the set of taxa that you fed it. The database is created by and for a program named sqlite, which is a very handy, portable SQL database. For more info on SQL and SQLITE, see this sqlite-tutorial. I'll briefly cover the database contents and use below.

First, take a look at the contents of the database by running:

```
sqlite3 probe.matches.sqlite
```

You'll now see something like:

```
SQLite version 3.7.3
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

It's often easier to change some defaults for better viewing, so at the prompt, paste in the following:

```
sqlite> .mode columns
sqlite> .headers on
sqlite> .nullvalue .
```

Tip: For more info on sqlite "dot" commands, you can type `.help`.

Now that that's done, let's see which tables the database contains by running the `.tables` command:

```
sqlite> .tables
match_map  matches
```

This tells us there's two tables in the database, named `match_map` and `matches`.

The `matches` table

Let's take a look at the contents of the `matches` table. Once you've started the sqlite interface, run:

```
sqlite> SELECT * FROM matches LIMIT 10;
```

This query select all rows (`SELECT *`) from the `matches` table (`FROM matches`) and limits the number of returned rows to 10 (`LIMIT 10`). This will output data that look something like:

```
uce         genus_species1  genus_species2  genus_species3
----------  --------------  --------------  --------------
uce-500     1               .               .
uce-501     1               .               .
uce-502     1               .               .
uce-503     1               1               1
uce-504     1               .               .
uce-505     1               .               .
uce-506     .               .               .
uce-507     1               .               .
uce-508     1               1               .
uce-509     1               1               1
```

Basically, what this indicates is that you enriched 9 of 10 targeted UCE loci from `genus_species1`, 3 of 10 UCE loci in the list from `genus_species2`, and 2 of 10 UCE loci from `genus_species3`. The locus name is given in the `uce column`. Remember that we've limited the results to 10 rows for the sake of making the results easy to view.

If we wanted to see only those loci that enriched in all species, we could run:

```
sqlite> SELECT * FROM matches WHERE genus_species1 = 1
   ...> AND genus_species2 = 1 AND genus_species3 = 1;
```

Assuming we only had those 10 UCE loci listed above in the database, if we ran this query, we would see something like:

```
uce         genus_species1  genus_species2  genus_species3
----------  --------------  --------------  --------------
uce-503     1               1               1
uce-509     1               1               1
```

Basically, the `matches` table and this query are what we run to generate **complete** (only loci enriched in all taxa) and **incomplete** (all loci enriched from all taxa) matrices very easily and quickly (see *Creating a data matrix configuration file*).

The `match_map` table

The `match_map` table shows us which species-specific, contigs match which UCE loci. Because each assembly program assigns an arbitrary designator to each assembled contig, we need to map these arbitrary designators (which also differ for each taxon/OTU) to the UCE locus to which it corresponds. Because assembled contigs are also not in any particular orientation relative to each other across taxa/OTUs (i.e., they may be 5' - 3' or 3' - 5'), the database also records the orientation of all contigs relative to orientation of each probe in the probes file.

Let's take a quick look at the `match_map` table:

```
SELECT * FROM match_map LIMIT 10;
```

This query is similar to the one that we ran against `matches` and returns the first 10 rows of the `match_map` table:

```
uce         genus_species1  genus_species2  genus_species3
----------  --------------  --------------  --------------
uce-500     node_233(+)     .               .
uce-501     node_830(+)     .               .
uce-502     node_144(-)     .               .
uce-503     node_1676(+)    node_243(+)     node_322(+)
uce-504     node_83(+)      .               .
uce-505     node_1165(-)    .               .
uce-506     .               .               .
uce-507     node_967(+)     .               .
uce-508     node_671(+)     node_211(-)     .
uce-509     node_544(-)     node_297(+)     node_37(+)
```

As stated above, these results show which assembled contigs "hit" particular UCE loci. So, if we were to open the `$ASSEMBLY/contigs/genus_species1.contigs.fasta` symlink the contig named `node_1676` corresponds to UCE locus `uce-503`. Because contigs are named arbitrarily during assembly, this same UCE locus is also found in genus_species2, but it is named `node-243`.

Each entry in the rows also provides the orientation for particular contigs `(-)` or `(+)`. This orientation is relative to the orientation of the UCE probes/locus in the source genome (e.g., chicken for tetrapod probes).

We use this table to generate a FASTA file of UCE loci for alignment (see :ref :*fasta-file*), after we've identified the loci we want in a particular data set (see *Creating a data matrix configuration file*). The code for this step also uses the associated orientation data to ensure that all the sequence data have the same orientation prior to alignment (some aligners will force alignment of all reads using the given orientation rather than also trying the reverse complement and picking the better alignment of the two).

Now that we know the taxa for which we've enriched UCE loci and which contigs we've assembled match which UCE loci, we're ready to generate some data matrices.

The data matrix generation process consists of two distinct parts:

1. Getting locus counts and generating a taxon set
2. Extracting FASTA data from our `$ASSEMBLY/contigs` based on the taxon set

Creating a data matrix configuration file

After we identify the UCE loci we enriched, but before we extract fasta data from our `$ASSEMBLY/contigs` corresponding to those loci, we need to create a data matrix configuration file that denotes (1) which taxa we want to include in a given analysis and (2) which loci will be included with this taxon set.

The taxa included in the data matrix configuration file are determined by the user - you input a list of taxa you want to the analysis. The UCE loci included in the data matrix configuration are then determined by the software which compares the requested taxa to UCE match results in `probe.matches.sqlite` and two flags that you pass either one requesting **complete data matrix** or one requesting an **incomplete data matrix**.

complete matrix A phylogenetic matrix (typically sequence data) in which there are no missing data at any locus for any taxon/OTU.

incomplete matrix A phylogenetic matrix (typocally sequence data) in which data may be missing from a given taxon or a given loci (or both).

During the creation of the data matrix configuration file you can also include additional data from pre-existing UCE match databases and contigs (see :ref :*outgroup-data*).

We'll start very simply.

Complete taxon set

First, let's generate a data matrix configuration file from only the current UCE enrichments that will be **complete** - meaning that we will not include loci where certain taxa have no data (either the locus was not enriched for that taxon or removed during the filtering process for duplicate loci).

To do this, you need to create a starting taxon-configuration file (a text-based file) denoting the taxa we want in the data set. The taxon-configuration file should look exactly like this (substitute in your taxon names):

```
[dataset1]
genus_species1
genus_species2
genus_species3
```

Let's assume you save this file as `datasets.conf`. Now, to create the data matrix configuration file from this taxon-configuration file, run:

```
# create the output directory for this taxon set
mkdir /path/to/uce/taxon-set1/

# create the data matrix configuration file
phyluce_assembly_get_match_counts \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --taxon-list-config datasets.conf \
    --taxon-group 'dataset1' \
    --output /path/to/uce/taxon-set1/dataset1.conf
```

This will basically run a query against the database, and pull out those loci for those taxa in the *datasets.conf* file having UCE contigs.

Results

The output printed to the screen and `$LOG` file should look something like:

```
2014-04-24 17:25:08,145 - get_match_counts - INFO - =================== Starting get_
↪match_counts ===================
2014-04-24 17:25:08,145 - get_match_counts - INFO - Version: git 7aec8f1
2014-04-24 17:25:08,145 - get_match_counts - INFO - Argument --extend_locus_db: None
2014-04-24 17:25:08,145 - get_match_counts - INFO - Argument --incomplete_matrix:␣
↪False
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --keep_counts: False
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --locus_db: /path/to/uce/
↪output/probes.matches.sqlite
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --log_path: /path/to/uce
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --optimize: False
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --output: /path/to/uce/
↪taxon-set1/dataset1.conf
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --random: False
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --sample_size: 10
2014-04-24 17:25:08,146 - get_match_counts - INFO - Argument --samples: 10
2014-04-24 17:25:08,147 - get_match_counts - INFO - Argument --silent: False
2014-04-24 17:25:08,147 - get_match_counts - INFO - Argument --taxon_group: dataset1
2014-04-24 17:25:08,147 - get_match_counts - INFO - Argument --taxon_list_config:␣
↪datasets.conf
2014-04-24 17:25:08,147 - get_match_counts - INFO - Argument --verbosity: INFO
2014-04-24 17:25:08,150 - get_match_counts - INFO - There are 3 taxa in the taxon-
↪group '[dataset1]' in the config file dataset1.conf
2014-04-24 17:25:08,151 - get_match_counts - INFO - Getting UCE names from database
2014-04-24 17:25:08,407 - get_match_counts - INFO - There are 1314 total UCE loci in␣
↪the database
2014-04-24 17:25:11,046 - get_match_counts - INFO - Getting UCE matches by organism␣
↪to generate a COMPLETE matrix
2014-04-24 17:25:11,051 - get_match_counts - INFO - There are 306 shared UCE loci in␣
↪a COMPLETE matrix
2014-04-24 17:25:11,051 - get_match_counts - INFO -     Failed to detect 428 UCE loci␣
↪in genus_species1
2014-04-24 17:25:11,051 - get_match_counts - INFO -     Failed to detect 380 UCE loci␣
↪in genus_species2
2014-04-24 17:52:54,850 - get_match_counts - INFO - Writing the taxa and loci in the␣
↪data matrix to /path/to/uce/taxon-set1/dataset1.conf
2014-04-24 17:52:54,862 - get_match_counts - INFO - =================== Completed get_
↪match_counts ==================
```

This basically says that although we've detected a total of 1,314 UCE loci in the 3 taxa in which we are interested, when we boil those down to a complete matrix, the complete matrix is only going to contain 306 UCE loci (of the 1,314). We had to drop 428 loci because we did not detect them in genus_species1 and we had to drop another 380 loci because we did not detect them in genus_species2.

The output written to the `/path/to/uce/taxon-set1/dataset1.conf` will look something like:

```
[Organisms]
genus_species1
genus_species2
genus_species3
```

(continues on next page)

(continued from previous page)

```
[Loci]
uce-1005
uce-1018
uce-1025
uce-1028
uce-1042
uce-1055
uce-1060
uce-107
uce-1073
uce-1074
uce-1076
uce-108
...
```

Taxon set membership and locus number

Now, you might think that increasing the locus count is simply a matter of removing *genus_species1* from the list of taxa. This is not strictly true, however, given the vagaries of hits and misses among taxa. *phyluce_assembly_get_match_counts* has several other options to help you determine which taxa may be causing problems, but picking the best combination of taxa to give you the highest number of loci is a reasonably hard optimization problem.

Incomplete data matrix

You may not always want a complete data matrix. Or generating a complete matrix drops too many loci for your tastes (it often does). In that case, you can easily generate an incomplete dataset using the following:

```
# create the data matrix configuration file
phyluce_assembly_get_match_counts \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --taxon-list-config datasets.conf \
    --taxon-group 'dataset1' \
    --output /path/to/uce/taxon-set1/dataset1.conf \
    --incomplete-matrix
```

Attention: Note the addition of the `--incomplete-matrix` flag.

This will generate a dataset that includes any loci enriched across the taxa in the *datasets.conf* file.

Note: You do not determine the "completeness" of the final data matrix that you want to create during this stage - that happens later, after alignment (see *Finalize matrix completeness*). As a result, we are alinging data from any and all UCE loci having >= 3 taxa, which allows us to flexibly select the level of incompleteness later, without having to re-run our alignments.

Creating additional data matrix configuration files for other analyses

If you want to generate/evaluate many data matrix configuration files containing different taxa, you can simply create new lists within the *datasets.conf* file like so:

```
[dataset1]
genus_species1
genus_species2
genus_species3

[dataset2]
genus_species2
genus_species3
genus_species4
genus_species5
genus_species6
```

And then you can run `phyluce_assembly_get_match_counts` against this new section to output the data matrix configuration files:

```
# create the data matrix configuration file
phyluce_assembly_get_match_counts \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --taxon-list-config datasets.conf \
    --taxon-group 'dataset2' \
    --output /path/to/uce/taxon-set2/dataset2.conf
```

In this way, you can get some idea of how different taxon-set memberships affect the resulting data matrix configuration files *prior to* extracting the relevant FASTA data from `$ASSEMBLY/contigs` - which is a reasonably slow process.

Incorporating outgroup/other data

You may want to include outgroup data from another source into your datasets. This can be from the pre-processed outgroup data files, but it doesn't need to be these outgroup data. These additional data can also be contigs previously assembled from a different set of taxa.

Hint: **ADVANCED**: If you want to include outgroup data from genome-enabled taxa, we have already created several repositories containing these data. We maintaing these data under version control at: https://github.com/faircloth-lab/uce-probe-sets. To download these data and use them in your analyses, you can clone the data using git:

```
git clone https://github.com/faircloth-lab/uce-probe-sets
```

Then update your `--taxon-list-config` file and provide the proper paths to the cloned data, as detailed below.

The first step of this process is to setup your `--taxon-list-config` slightly differently - by indicating taxa from external data sources using asterisks:

```
[dataset3]
genus_species1
genus_species2
genus_species3
genus_species4*
genus_species5*
```

Here, `genus_species4*` and `genus_species5*` come from an external data source.

Then, you need to pass `phyluce_assembly_get_match_counts` the location of both **your** `--locus-db` and the `--extend-locus-db`. For example:

```
# create the data matrix configuration file
phyluce_assembly_get_match_counts \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --taxon-list-config datasets.conf \
    --taxon-group 'dataset3' \
    --extend-locus-db /path/to/some/other/probe.matches.sqlite \
    --output /path/to/uce/taxon-set3/dataset3.conf
```

To keep all this extension from getting too crazy, I've limited the ability to include external data to a single set. If you have lots of data from many different enrichments, you'll need to generate a *contigs* folder containing all these various assemblies (or symlinks to them), then align the probes to these data (see *Match contigs to probes*). Once you do that, you can extend your current data set with all of these other data.

Extracting FASTA data using the data matrix configuration file

Once we have created the data matrix configuration file containing data for our taxa of interest and those loci of interest, we need to extract the appropriate FASTA sequences from each assembly representing the taxon/OTU of interest (e.g. in `$ASSEMBLY/contigs`). This is a reasonably straightforward process that differs only slightly based on whether you are extracting a complete matrix of data, an incomplete matrix of data, and/or whether you are incorporating any external data sources.

Complete data matrix

To generate FASTA file containing the sequence data from a complete data matrix configuration file, run:

```
phyluce_assembly_get_fastas_from_match_counts \
    --contigs /path/to/assembly/contigs/ \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --match-count-output /path/to/uce/taxon-set1/dataset1.conf \
    --output /path/to/uce/taxon-set1/dataset1.fasta
```

Incomplete data matrix

Similarly, to generate a FASTA file containing the sequence data from a complete data matrix configuration file, run:

```
phyluce_assembly_get_fastas_from_match_counts \
    --contigs /path/to/assembly/contigs/ \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --match-count-output /path/to/uce/taxon-set/dataset1.conf \
    --incomplete-matrix /path/to/uce/taxon-set1/dataset1.incomplete \
    --output /path/to/uce/taxon-set1/dataset1.fasta
```

Attention: Note the addition of the `--incomplete-matrix` option. This creates an output file that contains the names of the **missing** loci by taxon/OTU. You can name this file anything you like. I tend to use `.incomplete` as the extension so that it is clear what this file contains.

Incorporating outgroup/other data

When we're incorporating external data, we need to pass the name of the external database as well as the name of the external `contigs`. To generate a FASTA file containing the sequence data from a complete data matrix configuration that includes exeternal data sources, run:

```
phyluce_assembly_get_fastas_from_match_counts \
    --contigs /path/to/assembly/contigs/ \
    --locus-db /path/to/uce/output/probe.matches.sqlite \
    --match-count-output /path/to/uce/taxon-set1/dataset1.conf \
    --incomplete-matrix /path/to/uce/taxon-set1/dataset1.incomplete \
    --extend-locus-db /path/to/some/other/probe.matches.sqlite \
    --extend-locus-contigs /path/to/some/other/contigs \
    --output /path/to/uce/taxon-set3/dataset3.fasta
```

Aligning and trimming FASTA data

With all of that out of the way, things get much easier to deal with. Now, we need to align our data across loci, and once we're done with that, the remaining operations we can run on the data are format-conversions, QC steps, matrix trimming for completeness, and any number of other fun things.

Aligning the amount of data generated by enrichment approaches is reasonably computationally intensive - so the alignment step goes fastest if you have a multicore machine. You also have several alignment options available, although I would suggest sticking with MAFFT.

Attention: The alignment process, as implemented by phyluce, includes trimming steps that trim ragged edges and remove alignments that become to short following trimming.

To turn trimming off and trim alignments using another approach, pass the `--no-trim` option. There are also several more options related to trimming that you can tweak. To view these, run `phyluce_align_seqcap_align --help`.

Also see the *Tutorial I: UCE Phylogenomics* for more info on trimming options.

Complete data matrix

Alignment

To align the loci, by taxon, in the FASTA file you just created, run:

```
phyluce_align_seqcap_align \
    --fasta /path/to/uce/taxon-set1/dataset1.fasta \
    --output /path/to/uce/taxon-set1/mafft-nexus/ \
    --taxa 3 \
    --aligner mafft \
    --cores 8
```

Attention: If you pass more `--cores` than your machine has, you will receive an error.

Note: Here, we are accepting the default, output alignment format ("nexus"). To change that format to something else, pass the `--output-format` option with a choice of {fasta,nexus,phylip,clustal,emboss,stockholm}.

Alignment stats

Once you have alignments, it's nice to get a general sense of their length and composition. You can quickly (with a multicore machine) summarize thousands of alignments by running:

```
phyluce_align_get_align_summary_data \
    --alignments /path/to/uce/taxon-set1/mafft-nexus/ \
    --cores 12
```

This will produce output that looks similar to:

```
2014-04-24 17:31:15,724 - get_align_summary_data - INFO - ================ Starting␣
↪get_align_summary_data ================
2014-04-24 17:31:15,724 - get_align_summary_data - INFO - Version: git 7aec8f1
2014-04-24 17:31:15,724 - get_align_summary_data - INFO - Argument --alignments: /
↪path/to/uce/taxon-set1/mafft-nexus/
2014-04-24 17:31:15,724 - get_align_summary_data - INFO - Argument --cores: 12
2014-04-24 17:31:15,724 - get_align_summary_data - INFO - Argument --input_format:␣
↪nexus
2014-04-24 17:31:15,724 - get_align_summary_data - INFO - Argument --log_path: /path/
↪to/uce/taxon-set1/log
2014-04-24 17:31:15,725 - get_align_summary_data - INFO - Argument --show_taxon_
↪counts: False
2014-04-24 17:31:15,725 - get_align_summary_data - INFO - Argument --verbosity: INFO
2014-04-24 17:31:15,725 - get_align_summary_data - INFO - Getting alignment files
2014-04-24 17:31:15,729 - get_align_summary_data - INFO - Computing summary␣
↪statistics using 12 cores
2014-04-24 17:31:16,653 - get_align_summary_data - INFO - -----------------------␣
↪Alignment summary -----------------------
2014-04-24 17:31:16,654 - get_align_summary_data - INFO - [Alignments] loci:    306
2014-04-24 17:31:16,654 - get_align_summary_data - INFO - [Alignments] length:  223,
↪929
2014-04-24 17:31:16,654 - get_align_summary_data - INFO - [Alignments] mean:    731.79
2014-04-24 17:31:16,654 - get_align_summary_data - INFO - [Alignments] 95% CI:  17.01
2014-04-24 17:31:16,654 - get_align_summary_data - INFO - [Alignments] min:     275
2014-04-24 17:31:16,654 - get_align_summary_data - INFO - [Alignments] max:     1,109
2014-04-24 17:31:16,655 - get_align_summary_data - INFO - -------------------------␣
↪Taxon summary -------------------------
2014-04-24 17:31:16,655 - get_align_summary_data - INFO - [Taxa] mean:          27.00
2014-04-24 17:31:16,655 - get_align_summary_data - INFO - [Taxa] 95% CI:        0.00
2014-04-24 17:31:16,656 - get_align_summary_data - INFO - [Taxa] min:           27
2014-04-24 17:31:16,656 - get_align_summary_data - INFO - [Taxa] max:           27
2014-04-24 17:31:16,656 - get_align_summary_data - INFO - ----------------- Missing␣
↪data from trim summary ----------------
2014-04-24 17:31:16,656 - get_align_summary_data - INFO - [Missing] mean:       7.61
2014-04-24 17:31:16,656 - get_align_summary_data - INFO - [Missing] 95% CI:     0.24
2014-04-24 17:31:16,656 - get_align_summary_data - INFO - [Missing] min:        1.13
2014-04-24 17:31:16,657 - get_align_summary_data - INFO - [Missing] max:        15.79
2014-04-24 17:31:16,661 - get_align_summary_data - INFO - --------------------␣
↪Character count summary --------------------
2014-04-24 17:31:16,661 - get_align_summary_data - INFO - [All characters]      6,046,
↪083
```

(continues on next page)

(continued from previous page)

```
2014-04-24 17:31:16,661 - get_align_summary_data - INFO - [Nucleotides]        4,924,
↪129
2014-04-24 17:31:16,661 - get_align_summary_data - INFO - ---------------- Data␣
↪matrix completeness summary ---------------
2014-04-24 17:31:16,661 - get_align_summary_data - INFO - [Matrix 50%]         306␣
↪alignments
2014-04-24 17:31:16,661 - get_align_summary_data - INFO - [Matrix 55%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 60%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 65%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 70%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 75%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 80%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 85%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 90%]         306␣
↪alignments
2014-04-24 17:31:16,662 - get_align_summary_data - INFO - [Matrix 95%]         306␣
↪alignments
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - ------------------------␣
↪Character counts -----------------------
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - [Characters] '-' is present␣
↪651,009 times
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - [Characters] '?' is present␣
↪470,945 times
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - [Characters] 'A' is present␣
↪1,386,821 times
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - [Characters] 'C' is present␣
↪1,089,729 times
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - [Characters] 'G' is present␣
↪1,094,159 times
2014-04-24 17:31:16,663 - get_align_summary_data - INFO - [Characters] 'T' is present␣
↪1,353,420 times
2014-04-24 17:31:16,664 - get_align_summary_data - INFO - ================ Completed␣
↪get_align_summary_data ===============
```

Locus name removal

For historical reasons, and also for users to ensure that the sequence data aligned together are from the same loci, each sequence line in the alignment file output by `seqcap_align` contains the `genus_species1` designator, but the `genus_species1` designator is also prepended with the locus name (e.g. `uce-1005_genus_species1`). We need to remove these if we plan to concatenate the loci (*RAxML*). More generally, it is a good idea to remove locus names from sequence lines before running any analyses. To do this, run:

```
phyluce_align_remove_locus_name_from_files \
    --alignments /path/to/uce/taxon-set1/mafft-nexus/ \
    --output /path/to/uce/taxon-set1/mafft-nexus-clean/ \
    --taxa 3
```

Incomplete data matrix

Alignment

The only difference for an alignment of incomplete data is that we also pass the `--incomplete-matrix` flag, which tells the code to expect that some loci will not contain data across all taxa:

```
phyluce_align_seqcap_align \
    --fasta /path/to/uce/taxon-set2/dataset2.fasta \
    --output /path/to/uce/taxon-set2/mafft-nexus/ \
    --taxa 34 \
    --aligner mafft \
    --cores 12 \
    --incomplete-matrix
```

Alignment stats

Once you have alignments, it's nice to get a general sense of their length and composition. You can quickly (with a multicore machine) summarize thousands of alignments by running:

```
phyluce_align_get_align_summary_data \
    --alignments /path/to/uce/taxon-set1/mafft-nexus/ \
    --cores 12
```

This will produce output that looks similar to:

```
2014-04-24 20:11:18,208 - get_align_summary_data - INFO - ================ Starting␣
↪get_align_summary_data ================
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Version: git 7aec8f1
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Argument --alignments: /
↪path/to/uce/taxon-set1/mafft-nexus/
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Argument --cores: 12
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Argument --input_format:␣
↪nexus
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Argument --log_path: /path/
↪to/uce/taxon-set1/log
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Argument --show_taxon_
↪counts: False
2014-04-24 20:11:18,209 - get_align_summary_data - INFO - Argument --verbosity: INFO
2014-04-24 20:11:18,210 - get_align_summary_data - INFO - Getting alignment files
2014-04-24 20:11:18,253 - get_align_summary_data - INFO - Computing summary␣
↪statistics using 12 cores
2014-04-24 20:11:20,573 - get_align_summary_data - INFO - -----------------------␣
↪Alignment summary -----------------------
2014-04-24 20:11:20,574 - get_align_summary_data - INFO - [Alignments] loci:    1,104
2014-04-24 20:11:20,574 - get_align_summary_data - INFO - [Alignments] length:  752,
↪617
2014-04-24 20:11:20,574 - get_align_summary_data - INFO - [Alignments] mean:    681.72
2014-04-24 20:11:20,574 - get_align_summary_data - INFO - [Alignments] 95% CI:  13.03
2014-04-24 20:11:20,574 - get_align_summary_data - INFO - [Alignments] min:     169
2014-04-24 20:11:20,574 - get_align_summary_data - INFO - [Alignments] max:     4,520
2014-04-24 20:11:20,576 - get_align_summary_data - INFO - -------------------------␣
↪Taxon summary -------------------------
2014-04-24 20:11:20,576 - get_align_summary_data - INFO - [Taxa] mean:          24.29
2014-04-24 20:11:20,576 - get_align_summary_data - INFO - [Taxa] 95% CI:        0.26
```

(continues on next page)

(continued from previous page)

```
2014-04-24 20:11:20,576 - get_align_summary_data - INFO - [Taxa] min:           3
2014-04-24 20:11:20,576 - get_align_summary_data - INFO - [Taxa] max:           27
2014-04-24 20:11:20,577 - get_align_summary_data - INFO - ----------------- Missing␣
↪data from trim summary ----------------
2014-04-24 20:11:20,577 - get_align_summary_data - INFO - [Missing] mean:       7.97
2014-04-24 20:11:20,577 - get_align_summary_data - INFO - [Missing] 95% CI:     0.16
2014-04-24 20:11:20,578 - get_align_summary_data - INFO - [Missing] min:        0.44
2014-04-24 20:11:20,578 - get_align_summary_data - INFO - [Missing] max:        19.71
2014-04-24 20:11:20,592 - get_align_summary_data - INFO - --------------------␣
↪Character count summary --------------------
2014-04-24 20:11:20,592 - get_align_summary_data - INFO - [All characters]      18,
↪541,550
2014-04-24 20:11:20,592 - get_align_summary_data - INFO - [Nucleotides]         14,
↪713,956
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - ---------------- Data␣
↪matrix completeness summary ---------------
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 50%]          1048␣
↪alignments
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 55%]          1044␣
↪alignments
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 60%]          1035␣
↪alignments
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 65%]          1027␣
↪alignments
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 70%]          1024␣
↪alignments
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 75%]          1010␣
↪alignments
2014-04-24 20:11:20,594 - get_align_summary_data - INFO - [Matrix 80%]          998␣
↪alignments
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - [Matrix 85%]          994␣
↪alignments
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - [Matrix 90%]          906␣
↪alignments
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - [Matrix 95%]          794␣
↪alignments
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - ------------------------␣
↪Character counts -----------------------
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - [Characters] '-' is present␣
↪2,301,454 times
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - [Characters] '?' is present␣
↪1,526,140 times
2014-04-24 20:11:20,595 - get_align_summary_data - INFO - [Characters] 'A' is present␣
↪4,092,085 times
2014-04-24 20:11:20,596 - get_align_summary_data - INFO - [Characters] 'C' is present␣
↪3,267,550 times
2014-04-24 20:11:20,596 - get_align_summary_data - INFO - [Characters] 'G' is present␣
↪3,286,742 times
2014-04-24 20:11:20,596 - get_align_summary_data - INFO - [Characters] 'T' is present␣
↪4,067,579 times
2014-04-24 20:11:20,596 - get_align_summary_data - INFO - ================ Completed␣
↪get_align_summary_data ===============
```

Note: The alignment summary stats give you some idea of data matrix composition at varying levels of completeness in the `Data matrix completeness summary` section.

Locus name removal

For historical reasons, and also for users to ensure that the sequence data aligned together are from the same loci, each sequence line in the alignment file output by `seqcap_align` contains the `genus_species1` designator, but the `genus_species1` designator is also prepended with the locus name (e.g. `uce-1005_genus_species1`). We need to remove these if we plan to concatenate the loci (*RAxML*). More generally, it is a good idea to remove locus names from sequence lines before running any analyses. To do this, run:

```
phyluce_align_remove_locus_name_from_files \
    --alignments /path/to/uce/taxon-set1/mafft-nexus/ \
    --output /path/to/uce/taxon-set1/mafft-nexus-clean/ \
    --cores 12
```

Finalize matrix completeness

After checking the resulting alignment summary stats and checking your alignments for quality, you will generally want to cull the data set to reach your desired level of completeness. That is easily done by running the following, while inputting the set of alignments just generated using:

```
# the integer following --taxa is the number of TOTAL taxa
phyluce_align_get_only_loci_with_min_taxa \
    --alignments /path/to/uce/taxon-set1/mafft-nexus-clean/ \
    --taxa 34 \
    --percent 0.75 \
    --output /path/to/uce/taxon-set1/mafft-nexus-min-25-taxa/ \
    --cores 12
```

Attention: This program computes the `floor(taxa * percent)` and uses the resulting number to determine the `min(taxa)` allowed in an alignment of `--percent` completeness.

This will produce output that looks similar to:

```
2014-04-24 20:12:33,386 - get_only_loci_with_min_taxa - INFO - ==============␣
↪Starting get_only_loci_with_min_taxa =============
2014-04-24 20:12:33,387 - get_only_loci_with_min_taxa - INFO - Version: git 7aec8f1
2014-04-24 20:12:33,387 - get_only_loci_with_min_taxa - INFO - Argument --alignments:␣
↪/path/to/uce/taxon-set1/mafft-nexus
2014-04-24 20:12:33,387 - get_only_loci_with_min_taxa - INFO - Argument --cores: 12
2014-04-24 20:12:33,387 - get_only_loci_with_min_taxa - INFO - Argument --input_
↪format: nexus
2014-04-24 20:12:33,387 - get_only_loci_with_min_taxa - INFO - Argument --log_path:␣
↪None
2014-04-24 20:12:33,387 - get_only_loci_with_min_taxa - INFO - Argument --output: /
↪path/to/uce/taxon-set1/mafft-nexus-min-25-taxa
2014-04-24 20:12:33,388 - get_only_loci_with_min_taxa - INFO - Argument --percent: 0.
↪75
2014-04-24 20:12:33,388 - get_only_loci_with_min_taxa - INFO - Argument --taxa: 27
2014-04-24 20:12:33,388 - get_only_loci_with_min_taxa - INFO - Argument --verbosity:␣
↪INFO
2014-04-24 20:12:33,388 - get_only_loci_with_min_taxa - INFO - Getting alignment files
2014-04-24 20:12:35,293 - get_only_loci_with_min_taxa - INFO - Copied 1010 alignments␣
↪of 1104 total containing >= 0.75 proportion of taxa (n = 20)
2014-04-24 20:12:35,294 - get_only_loci_with_min_taxa - INFO - =============␣
↪Completed get_only_loci_with_min_taxa =============
```

(continues on next page)

(continued from previous page)

Add missing data designators

Sometimes, depending on how you will handle the alignments, you will need to add missing data designators for taxa missing from each alignment of a given locus. This will basically allow you to generate concatenated data sets and it may reduce error messages from other programs about files having unequal numbers of taxa. To do this, run:

```
phyluce_align_add_missing_data_designators \
    --alignments /path/to/uce/taxon-set1/mafft-nexus-min-25-taxa \
    --output /path/to/uce/taxon-set1/mafft-nexus-min-25-taxa \
    --match-count-output /path/to/uce/taxon-set/dataset1.conf \
    --incomplete-matrix /path/to/uce/taxon-set1/dataset1.incomplete \
    --log-path log \
    --cores 12
```

Note: Here, we're inputting the `--match-count-output` and the `--incomplete-matrix` we created earlier in the *Incomplete data matrix* and *Extracting FASTA data using the data matrix configuration file* sections.

Operations on alignments

Many workflows for phylogenetics simply involve converting one alignment format to another or changing something about the contents of a given alignment. We use many of these manipulations in the next section (see *Preparing concatenated alignment data for analysis*), as well.

Converting one alignment format to another

To convert one alignment type (e.g., nexus) to another (e.g., fasta), we have a relative simple bit of code to achieve that process. You can greatly speed this processing step up on a multicore machine with the `--cores` option:

```
phyluce_align_convert_one_align_to_another \
    --alignments /path/to/uce/taxon-set1/mafft-nexus \
    --output /path/to/uce/taxon-set1/mafft-fasta \
    --input-format nexus \
    --output-format fasta \
    --cores 8 \
    --log-path log
```

You can convert from/to:

1. fasta
2. nexus
3. phylip
4. phylip-relaxed (probably the one you want)
5. clustal
6. emboss
7. stockholm

Shortening taxon names

You can shorten taxon names (e.g. for use with strict phylip) by modifying the above command slightly to add `--shorten-names`:

```
phyluce_align_convert_one_align_to_another \
    --alignments /path/to/uce/taxon-set1/mafft-nexus \
    --output /path/to/uce/taxon-set1/mafft-fasta-shortnames \
    --input-format nexus \
    --output-format fasta \
    --cores 8 \
    --shorten-names \
    --log-path log
```

Excluding loci or taxa

You may want to exclude loci less than a certain length or having fewer than a particular number of taxa, or only containing certain taxa. You can accomplish that using:

```
phyluce_align_filter_alignments \
    --alignments /path/to/uce/taxon-set1/mafft-nexus \
    --output /path/to/a/new/directory \
    --input-format nexus \
    --containing-data-for genus_species1 genus_species2 \
    --min-length 100 \
    --min-taxa 5 \
    --log-path log
```

This will filter alignments that do not contain the taxa requested, those alignments shorter than 100 bp, and those alignments having fewer than 5 taxa (taxa with only missing data are not counted).

Extracting taxon data from alignments

Sometimes you may have alignments from which you want to extract data from a given taxon, format the alignment string as fasta, and do something with the fasta results:

```
phyluce_align_extract_taxon_fasta_from_alignments \
    --alignments /path/to/uce/taxon-set1/mafft-nexus \
    --taxon genus_species1 \
    --output /path/to/output/file.fasta
```

Preparing concatenated alignment data for analysis

Formatting data for analysis generally involves slight differences from the steps described above. There are several application-specific programs in phyluce.

RAxML

For RAxML, you need a concatenated phylip file. This is pretty easily created if you have an input directory of nexus alignments. To create a concatenated phylip file from many input alignments, run:

```
phyluce_align_concanatenate_alignments \
    --alignments /path/to/uce/taxon-set1/mafft-nexus \
    --output /path/to/uce/taxon-set1/mafft-raxml \
    --phylip
```

This will output a concatenated file named `mafft-raxml.phylip` in `/path/to/uce/taxon-set1/mafft-raxml`. It will also include a charset file, `mafft-raxml.charsets`.

MrBayes (Nexus format)

You can create a Nexus-formatted file for programs like MrBayes and Paup with:

```
phyluce_align_concanatenate_alignments \
    --alignments /path/to/uce/taxon-set1/mafft-nexus \
    --output /path/to/uce/taxon-set1/mafft-raxml \
    --nexus
```

The charsets will be included in the Nexus file.

3.4.4 Workflows

As of phyluce 1.7.0, there is new functionality that uses "workflows" to perform different actions. Key among these are things like computing coverage across UCE loci and phasing SNPs within UCE loci. These workflows use **Snakemake_**, internally, and they are pretty easily portable and/or easy to modify, if desired.

What's Different

Previously, phyluce used its own, internal pipeline code to run multi-step, bioinformatic workflows. These have now been moved into "workflows", which accomplish the same general steps but are much easier to maintain and run using **Snakemake_**.

Workflow Location

The workflow **Snakemake_** files should be packaged into your conda installation, in case you are interested in modifying them for any reason. To most easily find their location, activate the `phyluce` environment, then run:

```
# get location of python in our conda environment
which python

# this returns something like:
/Users/bcf/miniconda3/envs/phyluce/bin/python
```

This means that the `workflow` **Snakemake_** files will be located at `/Users/bcf/miniconda3/envs/phyluce/phyluce/workflows`. Individual workflows can be run directly by **Snakemake_** from this directory, or they can be copied elsewhere, modified, and run by **Snakemake_**. You can also run these workflows within phyluce (see below).

Workflow Configuration

The workflow configuration files are detailed below, but it's important to note that they use a **different** configuration format than other phyluce configuration files. Instead of Windows INI based format, the new workflows (and **Snakemake_**, in general) use YAML syntax. See examples below.

Different Workflows

Mapping

Right now, the "mapping" workflow precedes all other workflows and is responsible for mapping reads to contigs, marking duplicates, computing coverage, and outputting BAM files representing the mapped reads. In order to run this new workflow, create a YAML-formatted configuration file that contains the names and paths (relative or absolute) to your contigs and your trimmed reads:

```
reads:
    alligator-mississippiensis: ../../phyluce/tests/test-expected/raw-reads/alligator-
↪mississippiensis/
    gallus-gallus: ../../phyluce/tests/test-expected/raw-reads/gallus-gallus
    peromyscus-maniculatus: ../../phyluce/tests/test-expected/raw-reads/peromyscus-
↪maniculatus
    rana-sphenocephafa: ../../phyluce/tests/test-expected/raw-reads/rana-sphenocephafa

contigs:
    alligator-mississippiensis: ../../phyluce/tests/test-expected/spades/contigs/
↪alligator_mississippiensis.contigs.fasta
    gallus-gallus: ../../phyluce/tests/test-expected/spades/contigs/gallus_gallus.
↪contigs.fasta
    peromyscus-maniculatus: ../../phyluce/tests/test-expected/spades/contigs/
↪peromyscus_maniculatus.contigs.fasta
    rana-sphenocephafa: ../../phyluce/tests/test-expected/spades/contigs/rana_
↪sphenocephafa.contigs.fasta
```

The first section of the file gives the name and path to a folder of raw-reads for each sample (this folder is what results from illumiprocessor). The second section gives the name and path to the contigs assembled for each organism.

To map these reads to the assembled contigs, run:

```
phyluce_workflow --config <path to your config file> \
    --output <path to some output folder name> \
    --workflow mapping \
    --cores 1
```

This will run the workflow, and your results will end up in the output folder specified. The structure of the output folder will look something like the following:

```
.
+-- coverage
|   +-- all-taxon.summary.csv
|   +-- alligator-mississippiensis.samtools.cov.tdt
|   +-- alligator-mississippiensis.summary.csv
|   +-- gallus-gallus.samtools.cov.tdt
|   +-- gallus-gallus.summary.csv
|   +-- peromyscus-maniculatus.samtools.cov.tdt
|   +-- peromyscus-maniculatus.summary.csv
|   +-- rana-sphenocephafa.samtools.cov.tdt
```

(continues on next page)

(continued from previous page)

```
│   +── rana-sphenocephafa.summary.csv
+── mapped_reads
│   +── alligator-mississippiensis.fxm.sorted.md.bam
│   +── alligator-mississippiensis.fxm.sorted.md.bam.flagstats.txt
│   +── gallus-gallus.fxm.sorted.md.bam
│   +── gallus-gallus.fxm.sorted.md.bam.flagstats.txt
│   +── peromyscus-maniculatus.fxm.sorted.md.bam
│   +── peromyscus-maniculatus.fxm.sorted.md.bam.flagstats.txt
│   +── rana-sphenocephafa.fxm.sorted.md.bam
│   +── rana-sphenocephafa.fxm.sorted.md.bam.flagstats.txt
+── references
    +── alligator-mississippiensis.contigs.fasta
    +── alligator-mississippiensis.contigs.fasta.amb
    +── alligator-mississippiensis.contigs.fasta.ann
    +── alligator-mississippiensis.contigs.fasta.bwt
    +── alligator-mississippiensis.contigs.fasta.pac
    +── alligator-mississippiensis.contigs.fasta.sa
    +── gallus-gallus.contigs.fasta
    +── gallus-gallus.contigs.fasta.amb
    +── gallus-gallus.contigs.fasta.ann
    +── gallus-gallus.contigs.fasta.bwt
    +── gallus-gallus.contigs.fasta.pac
    +── gallus-gallus.contigs.fasta.sa
    +── peromyscus-maniculatus.contigs.fasta
    +── peromyscus-maniculatus.contigs.fasta.amb
    +── peromyscus-maniculatus.contigs.fasta.ann
    +── peromyscus-maniculatus.contigs.fasta.bwt
    +── peromyscus-maniculatus.contigs.fasta.pac
    +── peromyscus-maniculatus.contigs.fasta.sa
    +── rana-sphenocephafa.contigs.fasta
    +── rana-sphenocephafa.contigs.fasta.amb
    +── rana-sphenocephafa.contigs.fasta.ann
    +── rana-sphenocephafa.contigs.fasta.bwt
    +── rana-sphenocephafa.contigs.fasta.pac
    +── rana-sphenocephafa.contigs.fasta.sa
```

Within the `coverage` directory are outputs on a per-sample and overall basis. For example, `alligator-mississippiensis.summary.csv` will contain summary info on coverage for the `alligator-mississippiensis` contigs - one line for each contig. Overall summary statistics (by taxon) will be in `all-taxon.summary.csv`. BAM files resulting from the mapping are in the `mapped-reads` directory, along with the output of samtools `flagstats` for each BAM. The `references` directory contains the FASTA-formatted contigs you started with and their bwa indexes.

Attention: If you want to compute coverage on UCE contigs (only) versus all contigs that were assembled, run the probe/bait to contig matching, create a monolithic FASTA for whatever samples you want, explode that FASTA `--by-taxon`, then use the path to those files for each taxon in the `contig` section of the workflow config file, described above.

You can also perform a dry-run of the software by adding the `--dry-run` parameter, like so:

```
phyluce_workflow --config <path to your config file> \
    --output <path to some output folder name> \
    --workflow mapping \
    --cores 1 \
    --dry-run
```

This will show you what should happen, without performing the analysis. Log files from the **Snakemake_** run will be present in a hidden directory in your output folder named `.snakemake`. Like so:

```
.
+── .snakemake
│   +── auxiliary
│   +── conda
│   +── conda-archive
│   +── incomplete
│   +── locks
│   +── log
│   │   +── 2021-03-01T150829.811458.snakemake.log
│   +── metadata
│   +── scripts
│   +── shadow
│   +── singularity
+── coverage
+── mapped_reads
+── references
```

Phasing

The phasing workflow is a re-implementation of the approach that we used in Andermann et al. 2018 that uses mapping information (generated above), along with samtools and **pilon_** to output the phased contigs. The goal of reimplmentation was to make this pipeline more robust. You run the pipeline by (1) running the `mapping` workflow, above. Then, (2) you create a second configuration file that looks like the following:

```
bams:
    alligator-mississippiensis: ../tests/test-data/bams/alligator-mississippiensis.
↪fxm.sorted.md.bam
    gallus-gallus: ../tests/test-data/bams/gallus-gallus.fxm.sorted.md.bam
    peromyscus-maniculatus: ../tests/test-data/bams/peromyscus-maniculatus.fxm.sorted.
↪md.bam
    rana-sphenocephafa: ../tests/test-data/bams/rana-sphenocephafa.fxm.sorted.md.bam

contigs:
    alligator-mississippiensis: ../tests/test-data/contigs/alligator_mississippiensis.
↪contigs.fasta
    gallus-gallus: ../tests/test-data/contigs/gallus_gallus.contigs.fasta
    peromyscus-maniculatus: ../tests/test-data/contigs/peromyscus_maniculatus.contigs.
↪fasta
    rana-sphenocephafa: ../tests/test-data/contigs/rana_sphenocephafa.contigs.fasta
```

This contains a section pointing to the location of the BAM files created during `mapping`, and you can copy over the `contigs` section of the `mapping` config file. Finally, (3) you run the workflow with:

```
phyluce_workflow --config <path to your config file> \
    --output <path to some output folder name> \
    --workflow mapping \
    --cores 1
```

This produces a folder of output containing BAMs and FASTAs for each haplotye that looks like the following (here, only showing the results for `gallus-gallus` versus all 4 taxa in the configuration file:

```
.
+── bams
```

(continues on next page)

(continued from previous page)

```
|   +-- gallus-gallus.0.bam
|   +-- gallus-gallus.0.bam.bai
|   +-- gallus-gallus.1.bam
|   +-- gallus-gallus.1.bam.bai
|   +-- gallus-gallus.chimera.bam
+-- fastas
    +-- gallus-gallus.0.changes
    +-- gallus-gallus.0.fasta
    +-- gallus-gallus.0.vcf
    +-- gallus-gallus.1.changes
    +-- gallus-gallus.1.fasta
    +-- gallus-gallus.1.vcf
```

Right now, what you do with these files is left up to you (e.g. in terms of merging their contents and getting the data aligned). You can essentially group all the `*.0.fasta` and `*.1.fasta` files for all taxa together as new "assemblies" of data and start the phyluce analysis process over from `phyluce_assembly_match_contigs_to_probes`.

Correction

This is a new workflow that we've put together that helps account for sequencing depth and base-calling quality in assembled contigs. Essentially, you can think of this "correction" process as a filter that helps remove low-depth, low-quality base calls from your assembly data generated by phyluce. We are using this, in particular, with UCE data collected from toepads.

To run the workflow, (1) first run the mapping workflow above and (2) create a configuration file that looks like:

```
bams:
    alligator-mississippiensis: ../tests/test-data/bams/alligator-mississippiensis.
↪fxm.sorted.md.bam
    gallus-gallus: ../tests/test-data/bams/gallus-gallus.fxm.sorted.md.bam
    peromyscus-maniculatus: ../tests/test-data/bams/peromyscus-maniculatus.fxm.sorted.
↪md.bam
    rana-sphenocephafa: ../tests/test-data/bams/rana-sphenocephafa.fxm.sorted.md.bam

contigs:
    alligator-mississippiensis: ../tests/test-data/contigs/alligator_mississippiensis.
↪contigs.fasta
    gallus-gallus: ../tests/test-data/contigs/gallus_gallus.contigs.fasta
    peromyscus-maniculatus: ../tests/test-data/contigs/peromyscus_maniculatus.contigs.
↪fasta
    rana-sphenocephafa: ../tests/test-data/contigs/rana_sphenocephafa.contigs.fasta
```

This contains a section pointing to the location of the BAM files created during `mapping`, and you can copy over the `contigs` section of the `mapping` config file. Finally, (3) you run the workflow with:

```
phyluce_workflow --config <path to your config file> \
    --output <path to some output folder name> \
    --workflow correction \
    --cores 1
```

This produces a folder of output that looks like the following. Within this directory as a set of "consensus" contigs, where variant bases have been hard-masked that have `QUAL<20 | DP<5 | AN>2`:

```
.
+── consensus
│   +── alligator-mississippiensis.consensus.filt.fasta
│   +── gallus-gallus.consensus.filt.fasta
│   +── peromyscus-maniculatus.consensus.filt.fasta
│   +── rana-sphenocephafa.consensus.filt.fasta
+── filtered_norm_pileups
    +── alligator-mississippiensis.norm.flt-indels.Q20.DP10.bcf
    +── alligator-mississippiensis.norm.flt-indels.Q20.DP10.bcf.csi
    +── gallus-gallus.norm.flt-indels.Q20.DP10.bcf
    +── gallus-gallus.norm.flt-indels.Q20.DP10.bcf.csi
    +── peromyscus-maniculatus.norm.flt-indels.Q20.DP10.bcf
    +── peromyscus-maniculatus.norm.flt-indels.Q20.DP10.bcf.csi
    +── rana-sphenocephafa.norm.flt-indels.Q20.DP10.bcf
    +── rana-sphenocephafa.norm.flt-indels.Q20.DP10.bcf.csi
```

Once the "correction" process has been run, you can re-input the corrected contigs to the phyluce analysis process from the `phyluce_assembly_match_contigs_to_probes` program.

3.4.5 List of Phyluce Programs

Assembly

phyluce_assembly_assemblo_abyss

Assemble fastq data for phyluce using abyss.

phyluce_assembly_assemblo_spades

Assemble fastq data for phyluce using spades.

phyluce_assembly_assemblo_velvet

Assemble fastq data for phyluce using velvet.

phyluce_assembly_explode_get_fastas_file

Given an input "monolithic" fasta file of UCE contigs (from phyluce), break that file up into locus- or taxon-specific individual files.

phyluce_assembly_extract_contigs_to_barcodes

Takes as input the LOG file created during `phyluce_assembly_match_contigs_to_barcodes` (below) and outputs a more nicely-formatted table of results.

phyluce_assembly_get_bed_from_lastz

Given a lastz file produced by phyluce, convert those results to BED format.

phyluce_assembly_get_fasta_lengths

Given an input FASTA-formatted file, summarize the info on contigs within that file and output summary statistics.

phyluce_assembly_get_fastas_from_match_counts

Given a match-count file (produced from `phyluce_assembly_get_match_counts`), output a monolithic FASTA-formatted file of UCE loci.

phyluce_assembly_get_fastq_lengths

Given some input FASTQ data, output summary stastistics about those reads.

phyluce_assembly_get_match_counts

Given results from `phyluce_assembly_match_contigs_to_probes` (below) and a configuration file, output those taxa and loci for which matches exist in the UCE database. The config file looks like:

```
[all]
alligator_mississippiensis
anolis_carolinensis
gallus_gallus
mus_musculus
```

phyluce_assembly_match_contigs_to_barcodes

Given a directory of assembled contigs and a file containing an organismal barcode in FASTA format, check the contigs of all taxa in the directory for presence of the barcode sequence, extract that region of each contig for each taxon, and run the result against the BOLD database (for each taxon). Useful for checking species ID and also searching for potential contamination.

phyluce_assembly_match_contigs_to_probes

Given a directory of assembled contigs and a file of UCE baits/probes, search the contigs for those that match part/all of a given bait/probe at some level of stringency.

phyluce_assembly_screen_probes_for_dupes

Check a probe/bait file for potential duplicate baits/probes.

Alignment

phyluce_align_add_missing_data_designators

Sometimes alignments do not contain the same taxa as other alignments, and those “missing taxa” need to be added in. This program allows you to add those missing entries for the missing taxa, although this is often not needed when using the phyluce concatenation tools (see below), which automatically deal with this problem.

phyluce_align_concatenate_alignments

Given an input file of alignments, concatenate those alignments together and output either a `--nexus` or a `--phylip` formatted file, along with charset information (either as an extra file, for phylip, or within the `--nexus` formatted file).

phyluce_align_convert_degen_bases

If there are IUPAC degenerate base codes within an alignment, convert those to "N".

phyluce_align_convert_one_align_to_another

Convert alignments between formats. Can convert freely between FASTA, Nexus, Phylip, Phylip-relaxed, Clustal, Emboss, and Stockholm.

phyluce_align_explode_alignments

Given an input directry of alignments, "explode" those files into taxon- or locus-specific sequence files.

phyluce_align_extract_taxa_from_alignments

Given a set of alignments and a list of taxa to keep or remove from the alignments, make a new directory of alignments with those taxa kept or removed.

phyluce_align_extract_taxon_fasta_from_alignments

Given a set of alignments and a taxon to extract from them, extract the data for the taxon and format those data as a FASTA file.

phyluce_align_filter_alignments

Filter alignments having certain taxa or certain lengths and make a new directory without those alignments.

phyluce_align_format_concatenated_phylip_for_paml

This will convert a Phylip-formatted concatenated alignment for PAML's weird, internal format. Not sure if PAML needs this format any longer.

phyluce_align_get_align_summary_data

Given a directory of alignments, output summary statistics for those alignments quickly.

phyluce_align_get_gblocks_trimmed_alignments_from_untrimmed

Given a directory of alignments, use gblocks to trim alignment edges and output a new folder of trimmed alignments.

phyluce_align_get_incomplete_matrix_estimates

Given a directory of alignments, estimate the number of taxa present in various incomplete matrix scenarios.

phyluce_align_get_informative_sites

Given a directory of alignments, compute the number of informative sites. Can output a list of these, which can be used for additional filtering by # of sites.

phyluce_align_get_only_loci_with_min_taxa

Given a directory of alignments, filter those alignments for a minimum number of taxa, and output the filtered alignments to a new directory.

phyluce_align_get_ry_recoded_alignments

Given a directory of alignments, recode those as either "R/Y" or "0/1" and output the converted alignments to a new directory.

phyluce_align_get_smilogram_from_alignments

Given a directory of alignments, ouput a CSV-formatted file showing the number of sites in the alignment from the center to the edges. Can be input to R to make "smilogram" figures of UCE variation.

phyluce_align_get_taxon_locus_counts_in_alignments

Given a directory of alignments, get a count of taxa in each alignment. Can be used to filter alignments based on occupancy.

phyluce_align_get_trimal_trimmed_alignments_from_untrimmed

Given a directory of alignments, use trimAL to trim the alignments and write the trimmed alignments to a new directory.

phyluce_align_get_trimmed_alignments_from_untrimmed

Given a directory of alignments, use the phyluce edge-trimming algorithm to trim alignment edges and output a new folder of trimmed alignments.

phyluce_align_move_align_by_conf_file

Given an input configuration file, copy alignments present in the configuration file from one directory to a new directory. Useful for filtering alignments. The format of the config file looks like:

```
[all]
locus-name-1.nexus
locus-name-2.nexus
locus-name-3.nexus
```

And this will copy all three loci to the specified (new) directory.

phyluce_align_randomly_sample_and_concatenate

Given an input directory of alignments, randomly sample those alignments and output a file of those random sequences concatenated together.

phyluce_align_reduce_alignments_with_raxml

Given an input directory of alignments, use raxml to create "reduced" files of each alignment (where missing and low-info site patterns have been removed). Was needed to work around a pargenes bug but should not be necessary any longer.

phyluce_align_remove_empty_taxa

Given an input directory of alignments, remove those taxa within each alignment having no data.

phyluce_align_remove_locus_name_from_files

Given an input directory of alignments that still contain the names of each UCE loci along with each taxon, strip the name of the UCE locus from each taxon and output the result into a new directory.

phyluce_align_screen_alignments_for_problems

Given an input directory of alignments, screen those for problems such as weird nucleotide codes ("X") or runs of ambigious bases ("N").

phyluce_align_seqcap_align

Given a monolithic fasta file, align fasta sequences by locus and output the resulting alignments into a new directory.

phyluce_align_split_concat_nexus_to_loci

Given an input NEXUS-formatted file of a concatenated alignment (with charset info in the file), split the concatenated alignment back into component parts.

Genetrees

phyluce_genetrees_generate_multilocus_bootstrap_count

This is used with site and locus-based boostrap resampling. Not particularly recommended any longer.

phyluce_genetrees_get_mean_bootrep_support

Given a set of input genetrees, compute the mean bootrep support among those trees.

phyluce_genetrees_get_tree_counts

Given an input directory of alignments, uses the symmetric difference to count the number of similar and difference gene tree topologies.

phyluce_genetrees_rename_tree_leaves

Given an input tree and a config file detailing the mapping of old names to new names, convert the old leaf names in a tree to the new names in the config file. The config file format is similar to:

```
[standard]
acanthisitta_chloris_APP_002:Passeriformes__Acanthisittidae__acanthisitta_chloris_APP_
↪002__4763__1051
acrocephalus_arundinaceus_OUT_0054:Passeriformes__Acrocephalidae__acrocephalus_
↪arundinaceus_OUT_0054__4729__1062
aegithalos_caudatus_B10K_DU_002_10:Passeriformes__Aegithalidae__aegithalos_caudatus_
↪B10K_DU_002_10__4572__1038
aegotheles_bennettii_B10K_DU_029_76:Caprimulgiformes__Aegothelidae__aegotheles_
↪bennettii_B10K_DU_029_76__4806__1058
agapornis_roseicollis_OUT_0001:Psittaciformes__Psittaculidae__agapornis_roseicollis_
↪OUT_0001__4760__1040
agelaius_phoeniceus_OUT_0050:Passeriformes__Icteridae__agelaius_phoeniceus_OUT_0050__
↪4786__1060
```

Where the old name is on the left of the colon and the new name is on the right of the colon.

phyluce_genetrees_sort_multilocus_bootstraps

This is used with site and locus-based boostrap resampling. Not particularly recommended any longer.

NCBI

phyluce_ncbi_chunk_fasta_for_ncbi

Splits an input fasta file into chunks of 10,000 sequences because Sequin files should not contain more than 10,000 records.

phyluce_ncbi_prep_uce_align_files_for_ncbi

Given an input file of alignments, prep those for input to tbl2asn, which formats them for submission to NCBI. Also takes a config file with a format similar to what follows.

```
[exclude taxa]
ichthyopis kohtaoensis
plethodon chlorobryonis
phalacrocorax carbo
```

(continues on next page)

(continued from previous page)

```
[exclude loci]
uce-1809

[metadata]
molecule:DNA
moltype:genomic
location:genomic
note:ultra conserved element locus {}
specimen_voucher:{}

[vouchers]
Ardeotis kori:FLMNH 44254
Balaeniceps rex:LSUMZ B13372
Cathartes aura:LSUMZ B17242

[remap]
pterocles:pterocles exustus
zanclostomus javanicus:sphyrapicus varius
```

Probes

phyluce_probe_easy_lastz

Run an "easy" lastz search of one file against another file.

phyluce_probe_get_genome_sequences_from_bed

Given an input BED file, extracts fasta information matching the coordinates in the BED file.

phyluce_probe_get_locus_bed_from_lastz_files

Given a lastz results file where baits/probes were searched against a genome, output a BED-formatted file of the **locus** coodinates for each match of bait/baits to the genome.

phyluce_probe_get_multi_fasta_table

Make a table containing multi-way fasta information.

phyluce_probe_get_multi_merge_table

Make a table containing multi-way fasta information.

phyluce_probe_get_probe_bed_from_lastz_files

Given a lastz results file where baits/probes were searched against a genome, output a BED-formatted file of the **bait** coodinates for each match of bait/baits to the genome.

phyluce_probe_get_screened_loci_by_proximity

Given a FASTA file of properly formatted baits, keep only 1 locus (randomly) of those falling within a set distance from one another.

phyluce_probe_get_subsets_of_tiled_probes

Given a bait file that contains baits designed from multiple organisms, prune that bait file to contain only those baits from a desired subset of organisms.

phyluce_probe_get_tiled_probe_from_multiple_inputs

Design baits from multiple input genomes.

phyluce_probe_get_tiled_probes

Design baits from a single input genome.

phyluce_probe_query_multi_fasta_table

Query a multifasta table.

phyluce_probe_query_multi_merge_table

Query a multimerge table.

phyluce_probe_reconstruct_uce_from_probe

From a UCE bait set, reconstruct the UCE locus used for design.

phyluce_probe_remove_duplicate_hits_from_probes_using_lastz

Given a bait set, and some lastz results of matching that bait set to itself, screen those probes from the bait set that match other probes (these are putative duplicates).

phyluce_probe_remove_overlapping_probes_given_config

Given a config file, filter baits from a set that are in the config file.

phyluce_probe_run_multiple_lastzs_sqlite

Use phyluce to run multiple lastz searches across multiple input genomes.

phyluce_probe_slice_sequence_from_genomes

Given results from `phyluce_probe_run_multiple_lastzs_sqlite`, slice fasta sequences from the genomes where there were matches.

phyluce_probe_strip_masked_loci_from_set

Remove baits from a putative bait set design where bait sequences have a high degree of masking.

Utilities

phyluce_utilities_combine_reads

Combine groups of reads based on an input file in config format.

phyluce_utilities_filter_bed_by_fasta

Filter a BED file of UCEs given a FASTA file of UCEs.

phyluce_utilities_get_bed_from_fasta

Given an input fasta file of baits, prepared a BED-formatted file of their locations.

phyluce_utilities_merge_multiple_gzip_files

Merge together multiple gzip files from the same sample.

phyluce_utilities_merge_next_seq_gzip_files

Merge together multiple fastq gzip files from the next seq (these sometimes come as 4 files per sample).

phyluce_utilities_replace_many_links

Use a config file to reformat many symlinks all at once.

phyluce_utilities_sample_reads_from_files

Automatically randomly sample a fraction of reads from a fastq file.

phyluce_utilities_unmix_fasta_reads

Given an interleaved fastq file, convert that file into R1, R2, and singleton reads.

Workflow

phyluce_workflow

A single program to run a variety of **Snakemake_** workflows.

CHAPTER 4

Project info

4.1 Citing

If you use the phyluce code in any form, please cite the following manuscript:

If you are processing UCE data that you have collected by targeted enrichment using our probes/protocols, please cite the following manuscripts, which describes the first use of the general approach:

Then, if you are working in groups other than tetrapods, please cite the appropriate manuscript for the organisms you are working on.

4.1.1 References

Here are some additional references that have used UCE data for phylogenomic inference at both "deep" and "shallow" timescales. Most of these manuscripts used an older version of the code:

4.1.2 Other UCE References

4.2 License

4.2.1 Documentation

The documentation for phyluce is available under a CC-BY (2.0) license. This license gives you permission to copy, distribute, and trasmit the work as well as to adapt the work or use this work for commercial purposes, under the condition that you must attribute the work to the author(s).

If you use this documentation or the phyluce software for your own research, please cite both the software and (Faircloth et al. 2012). See the Citing section for more detail.

4.2.2 Software

4.3 Attributions

A large number of people have worked on different aspects of the UCE approach, including creating the laboratory methods to collect the data and the computational methods to analyze the data. Below, we have identified a list of approximately which people/groups did what.

4.3.1 Maintainer/Author

- Brant Faircloth (brant **at** faircloth-lab **dot** org)

4.3.2 Contributed to the code

- Nick Crawford (ngcrawford **at** gmail **dot** com)
- Jonathan Chang (me **at** jonathanchang **dot** org)
- Mike Harvey (mharve9 **at** lsu **dot** edu)
- Tobias Hofmann (tobiashofmann **at** gmx **dot** net)
- Carl Oliveros (carl **at** ku **dot** edu)

4.3.3 Developed the UCE approach

- Brant Faircloth (LSU)
- Travis Glenn (UGA)

4.3.4 Contributed to the UCE approach

- John McCormack (Occidental College)
- Robb Brumfield (LSU)
- Mike Alfaro (UCLA)
- Nick Crawford (Boston Univ.)
- Mike Harvey (LSU)
- Roger Nilsen (UGA)
- Brian Smith (LSU)
- Laurie Sorenson (LSU)
- Kevin Winker (U. Alaska - Fairbanks)

Contributed samples, funding, time, comments, etc.

Additionally, the following individuals have contributed samples, funding, laboratory work, computer code, documentation, or all of the above:

- Mike Braun (Smithsonian)
- Noor White (Smithsonian)
- Ed Braun (U. Florida)
- Rebecca Kimball (U. Florida)
- David Ray (MsState)
- Seán Brady (Smithsonian)
- Jesus Maldonado (Smithsonian)
- Jonathan Chang (UCLA)

4.4 Funding

4.4.1 Primary Sources

The National Science Foundation (NSF) and the Smithsonian Institution have supported a large portion of our work. The specific programs and proposal identifiers are below:

- NSF DEB-0841729
- NSF DEB-0956069
- NSF DEB-1242241
- NSF DEB-1242260
- NSF DEB-1242267
- Smithsonian Institution Consortium for Understanding and Sustaining a Biodiverse Planet

4.4.2 Secondary Sources

We have also received funding for computational support from and/or materials from the following organizations:

- Amazon Web Services (Education grants to BCF, NGC, JEM, and TCG)
- IDTDNA

4.5 Acknowledgements

We thank the following people, each of whom made contributions ensuring the success of our work. These include:

- Claire Mancuso
- Brant Peterson
- Brent Pederson
- Chris Moran
- Ken Jones
- Joe DeYoung
- LSU Genomics Facility
- UCLA Neuroscience Genomics Core
- M. Reasel

Bibliography

[BCF2015] Faircloth BC. 2016. PHYLUCE is a software package for the analysis of conserved genomic loci. Bioinformatics 32:786-788. doi:10.1093/bioinformatics/btv646.

[BCF2012] BC Faircloth, McCormack JE, Crawford NG, Harvey MG, Brumfield RT, Glenn TC. 2012. Ultraconserved elements anchor thousands of genetic markers spanning multiple evolutionary timescales. Systematic Biology 61: 717–726. doi:10.1093/sysbio/SYS004.

[JEM2012] McCormack JE, Faircloth BC, Crawford NG, Gowaty PA, Brumfield RT, Glenn TC. 2012. Ultraconserved elements are novel phylogenomic markers that resolve placental mammal phylogeny when combined with species tree analysis. Genome Res 26:746-754. doi:10.1101/gr.125864.111.

[NGC2012] Crawford NG, Faircloth BC, McCormack JE, Brumfield RT, Winker K, Glenn TC. 2012. More than 1000 ultraconserved elements provide evidence that turtles are the sister group of archosaurs. Biol Lett 8:783-786. doi:10.1098/rsbl.2012.0331.

[BCF2013] Faircloth BC, Sorenson L, Santini F, Alfaro ME. 2013. A phylogenomic perspective on the radiation of ray-finned fishes based upon targeted sequencing of ultraconserved elements. PlosONE 8:e65923. doi:10.1371/journal.pone.0065923.

[JEM2013] McCormack JE, Harvey MG, Faircloth BC, Crawford NG, Glenn TC, Brumfield RT. 2013. A phylogeny of birds based on over 1,500 loci collected by target enrichment and high-throughput sequencing. PlosOne 8:e54848. doi:10.1371/journal.pone.0054848.

[BTS2013] BT Smith, MG Harvey, BC Faircloth, TC Glenn, RT Brumfield. 2013. Target capture and massively parallel sequencing of ultraconserved elements (UCEs) for comparative studies at shallow evolutionary time scales. Syst Biol 63:83-95. doi:10.1093/sysbio/syt061.

[MGH2014] Sequence capture versus restriction site associated dna sequencing for phylogeography. MG Harvey, BT Smith, TC Glenn, BC Faircloth, RT Brumfield. arXiv:1312.6439.

[GB2004] Bejerano G, Pheasant M, Makunin I, Stephen S, Kent WJ, et al. (2004) Ultraconserved elements in the human genome. Science 304: 1321–1325. doi:10.1126/science.1098119.

[AS2004] Sandelin A, Bailey P, Bruce S, Engström PG, Klos JM, et al. (2004) Arrays of ultraconserved non-coding regions span the loci of key developmental genes in vertebrate genomes. BMC Genomics 5: 99. doi:10.1186/1471-2164-5-99.

[ED2005] Dermitzakis ET, Reymond A, Antonarakis SE (2005) Opinion: Conserved non-genic sequences — an unexpected feature of mammalian genomes. Nat Rev Genet 6:151–157. doi:10.1038/nrg1527.

[AS2005] Siepel A, Bejerano G, Pedersen JS, Hinrichs AS, Hou M, et al. (2005) Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. Genome Res 15: 1034–1050. doi:10.1101/gr.3715005.

[AW2005] Woolfe A, Goodson M, Goode D, Snell P, McEwen G, et al. (2005) Highly conserved non-coding sequences are associated with vertebrate development. PLoS Biol 3:116–130. doi:10.1371/journal.pbio.0030007.

[LP2006] Pennacchio LA, Ahituv N, Moses AM, Prabhakar S, Nobrega MA, et al. (2006) In vivo enhancer analysis of human conserved non-coding sequences. Nature 444: 499–502. doi:10.1038/nature05295.

[NA2007] Ahituv N, Zhu Y, Visel A, Holt A, Afzal V, et al. (2007) Deletion of Ultraconserved Elements Yields Viable Mice. PLoS Biol 5: e234. doi:10.1371/journal.pbio.0050234.

[WM2007] Miller W, Rosenbloom K, Hardison RC, Hou M, Taylor J, et al. (2007) 28-way vertebrate alignment and conservation track in the UCSC Genome Browser. Genome Res 17: 1797–1808. doi:10.1101/gr.6761107.

[AG2009] Gnirke A, Melnikov A, Maguire J, Rogov P, LeProust EM, et al. (2009) Solution hybrid selection with ultra-long oligonucleotides for massively parallel targeted sequencing. Nature Biotechnology 27: 182–189. doi:10.1038/nbt.1523.

[BB2010] Blumenstiel B, Cibulskis K, Fisher S, DeFelice M, Barry A, et al. (2010) Targeted exon sequencing by in-solution hybrid selection. Curr Protoc Hum Genet Chapter 18: Unit18.4. doi:10.1002/0471142905.hg1804s66.