Supplementary Information

Ava Hoffman, Jenny Cocciardi

Contents

1 About

This is supplementary material intended to be paired with files on GitHub at the repository https://github.com/avahoffman/urban-weed-genomics.

2 Introduction

In this experiment, we used quaddRAD library prep to prepare the sample DNA. This means that there were both two unique outer barcodes (typical Illumina barcodes) *AND* two unique inner barcodes (random barcode bases inside the adapters) for each sample - over 1700 to be exact!

The sequencing facility demultiplexes samples based on the outer barcodes (typically called 5nn and i7nn). Once this is done, each file still contains a mix of the inner barcodes. We will refer to these as "sublibraries" because they are sort of halfway demultiplexed. We separate them out bioinformatically later.

2.1 Raw Data File Naming - Sublibraries

Here's a bit of information on the file name convention. The typical raw file looks like this:

`AMH_macro_1_1_12px_S1_L001_R1_001.fastq.gz`

- These are author initials and "macro" stands for "Macrosystems". These are on every file.

  `AMH_macro`

- The first number is the *i5nn* barcode for the given sublibrary. We know all these samples have a i5nn barcode "1", so that narrows down what they can be. The second number is the *i7nn* barcode for the given sublibrary. We know all these samples have a i7nn barcode "1", so that further narrows down what they can be.

  `1_1`

- This refers to how many samples are in the sublibrary. "12px" means 12-plexed, or 12 samples. In other words, we will use the inner barcodes to further distinguish 12 unique samples in this sublibrary.

  `12px`

- This is a unique sublibrary name. S1 = 1 i5nn and 1 i7nn.

  `S1`

- This means this particular file came from lane 1 of the NovaSeq. There are four lanes. All samples should appear across all four lanes.

  `L001`

- This is the first (R1) of two paired-end reads (R1 and R2).

  `R1`

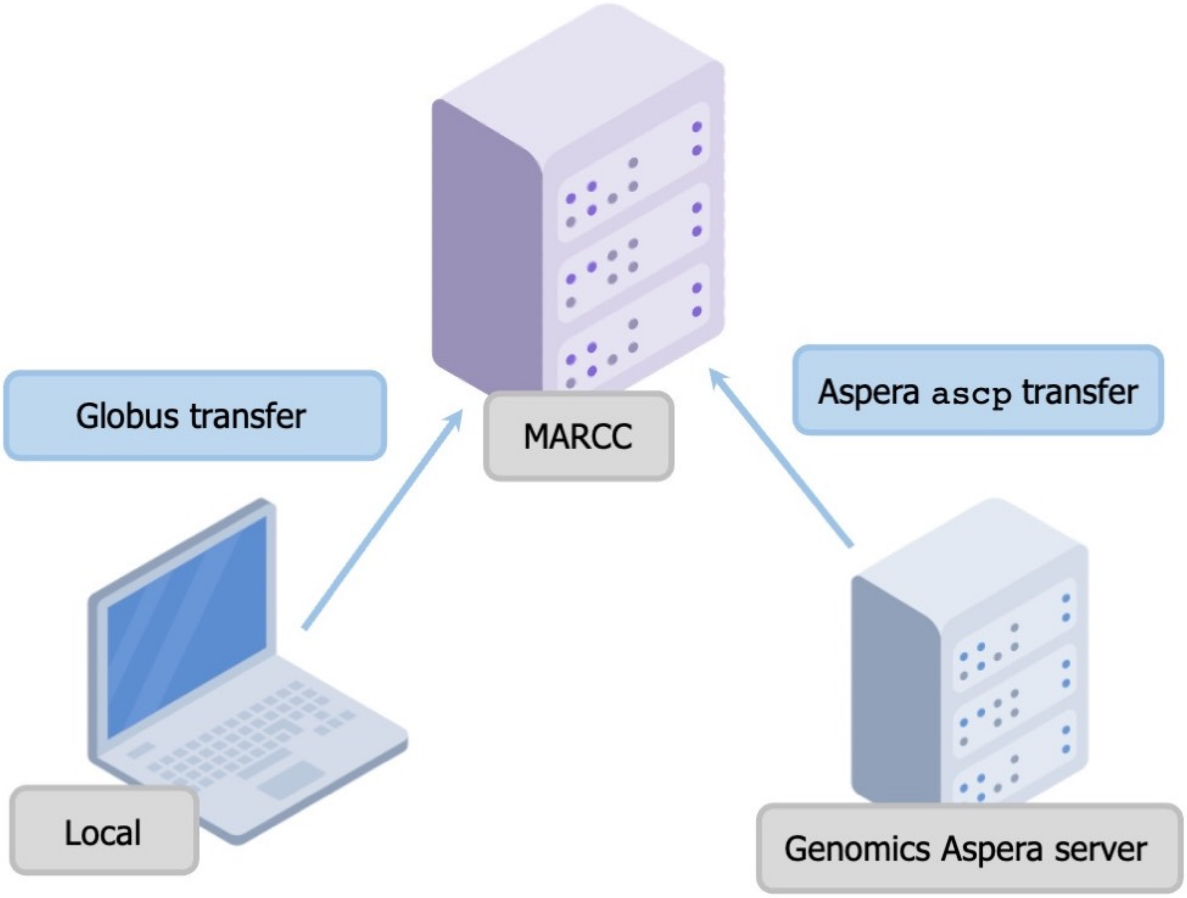
- The last part doesn't mean anything - it was just added automatically before the file suffix (`fastq.gz`)

  `001.fastq.gz`

2.2 A Note on File Transfers

There are three main systems at play for file transfer: the local machine, the sequencing facility's (GRCF) Aspera server, and MARCC. The Aspera server is where the data were/are stored immediately after sequencing. MARCC is where we plan to do preprocessing and analysis. Scripts and text files are easy for me to edit on my local machine. We used Globus to transfer these small files from my local machine to MARCC.

Midway through this analyses, we transitioned to another cluster, JHU's Rockfish. Scripts below, with the exception of file transfer from the Aspera server, should reflect the new filesystem, though you will have to adjust the file paths accordingly.



2.3 A Note on Species Names

Throughout this study, we examined 6 species. Sometimes we used abbreviations for easier file naming. These are:

1. *Cynodon dactylon*, "CD", or Bermuda grass
2. *Digitaria sanguinalis*, "DS", or crabgrass
3. *Erigeron canadensis*, "EC", or horseweed
4. *Lactuca serriola*, "LS", or prickly lettuce
5. *Poa annua*, "PA", or bluegrass
6. *Taraxacum officinale*, "TO", or dandelion

3 Preprocessing

3.1 Transfer Files

Referred to through files as "Step 1". Files can be found in the `01_transfer_files/` directory.

This directory contains files named in this convention: `01-aspera_transfer_n.txt`. These are text files containing the *names* of `fastq.gz` files that we wanted to transfer from the sequencing facility's Aspera server to the computing cluster (MARCC). This was to maximize ease of transferring only certain files over at once, since transferring could take a long time. We definitely did this piecemeal. Possible file names shown in Aspera Transfer File Names. There are multiple of these files so that we could parallelize (replace n with the correct number in the command used below). This text file will need to be uploaded to your scratch directory in MARCC.

Files were then transferred using the following commands. Before starting, make sure you are in a data transfer node. Then, load the aspera module. Alternatively, you can install the Aspera transfer software and use that.

```
module load aspera
```

Initiate the transfer from within your scratch directory:

```
ascp -T -l8G -i /software/apps/aspera/3.9.1/etc/asperaweb_id_dsa.openssh
--file-list=01-aspera_transfer_n.txt
--mode=recv --user=<aspera-user> --host=<aspera-IP> /scratch/users/<me>@jhu.edu
```

3.2 Concatenate Files and Install Stacks

Referred to through files as "Step 2". Files can be found in the `02_concatenate_and_check/` directory.

3.2.1 Concatenate Files for each Sublibrary

Step 2a. We ran my samples across the whole flow cell of the NovaSeq, so results came in 8 files for each demultiplexed sublibrary (4 lanes * paired reads). For example, for sublibrary 1_1, we'd see the following 8 files:

```
AMH_macro_1_1_12px_S1_L001_R1_001.fastq.gz
AMH_macro_1_1_12px_S1_L001_R2_001.fastq.gz
AMH_macro_1_1_12px_S1_L002_R1_001.fastq.gz
AMH_macro_1_1_12px_S1_L002_R2_001.fastq.gz
AMH_macro_1_1_12px_S1_L003_R1_001.fastq.gz
AMH_macro_1_1_12px_S1_L003_R2_001.fastq.gz
AMH_macro_1_1_12px_S1_L004_R1_001.fastq.gz
AMH_macro_1_1_12px_S1_L004_R2_001.fastq.gz
```

The `02_concatendate_and_check/02-concat_files_across4lanes.sh` script finds all files in the working directory with the name pattern `*_L001_*.fastq.gz` and then concatenates across lanes 001, 002, 003, and 004 so they can be managed further. The "L001" part of the filename is then eliminated. For example the 8 files above would become:

```
AMH_macro_1_1_12px_S1_R1.fastq.gz
AMH_macro_1_1_12px_S1_R2.fastq.gz
```

Rockfish uses slurm to manage jobs. To run the script, use the `sbatch` command. For example:

```
sbatch ~/code/02-concat_files_across4lanes.sh
```

This command will run the script from within the current directory, but will look for and pull the script from the code directory. This will concatenate all files within the current directory that match the loop pattern.

3.2.2 Download and Install Stacks

Step 2b. On Rockfish, Stacks will need to be downloaded to each user's code directory. Stacks, and software in general, should be compiled in an interactive mode or loaded via module. For more information on interactive mode, see `interact --usage`.

```
interact -p debug -g 1 -n 1 -c 1
module load gcc
```

Now download Stacks. We used version 2.60.

```
wget http://catchenlab.life.illinois.edu/stacks/source/stacks-2.60.tar.gz
tar xfvz stacks-2.60.tar.gz
```

Next, go into the stacks-2.60 directory and run the following commands:

```
./configure --prefix=/home/<your_username>/code4-<PI_username>
make
make install
export PATH=$PATH:/home/<your_username>/code4-<PI_username>/stacks-2.60
```

The filesystem patterns on your cluster might be different, and you should change these file paths accordingly.

3.3 Remove PCR Clones

Referred to through files as "Step 3". Files can be found in the `03_clone_filter/` directory.

3.3.1 Run PCR Clone Removal Script

Step 3a. The `03-clone_filter.sh` script runs `clone_filter` from Stacks. The program was run with options `--inline_inline --oligo_len_1 4 --oligo_len_2 4`. The `--oligo_len_x 4` options indicate the 4-base pair degenerate sequence was included on the outside of the barcodes for detecting PCR duplicates. The script uses the file name prefixes listed for each single sub-pooled library in `03-clone_filter_file_names.txt` and loops to run `clone_filter` on all of them. Possible file names shown in `clone_filter` File Names.

3.3.2 Parse PCR Clone Removal Results

Step 3b. If you want to extract descriptive statistics from the `clone_filter` output, you can use the `03.5-parse_clone_filter.py` script to do so. It can be run on your local terminal after transferring the `clone_filter.out` logs to your local computer.

```
source("03_clone_filter/examine_clones.R")
make_cloneplot()
```

3.4 Step 4 - Demultiplexing and Sample Filtering

Files can be found in the `04_demux_filter/` directory.

3.4.1 Step 4a - Demultiplex and Filter

The `04-process_radtags.sh` script runs `process_radtags` from Stacks. The program was run with options `-c -q --inline_inline --renz_1 pstI --renz_2 mspI --rescue --disable_rad_check`. The script uses the same file prefixes as Step 3 - `03-clone_filter.sh`. Each sub-pooled library has a forward and reverse read file that was filtered in the previous step. Like the above section, the script uses the file name prefixes listed for each single sub-pooled library in `04-process_radtags_file_names.txt` and loops to run `process_radtags` on all of them. Possible file names shown in `clone_filter` File Names.

Each sub-pooled library also has a demultiplexing file (`04-demux/` directory) that contains the sample names and inner(i5 and i7) barcodes. For example, the sublibrary 1_1, we'd see the following barcode file:

```
ATCACG  AGTCAA  DS.BA.PIK.U.1
CGATGT  AGTTCC  DS.BA.PIK.U.2
TTAGGC  ATGTCA  DS.BA.PIK.U.3
TGACCA  CCGTCC  DS.BA.PIK.U.4
```

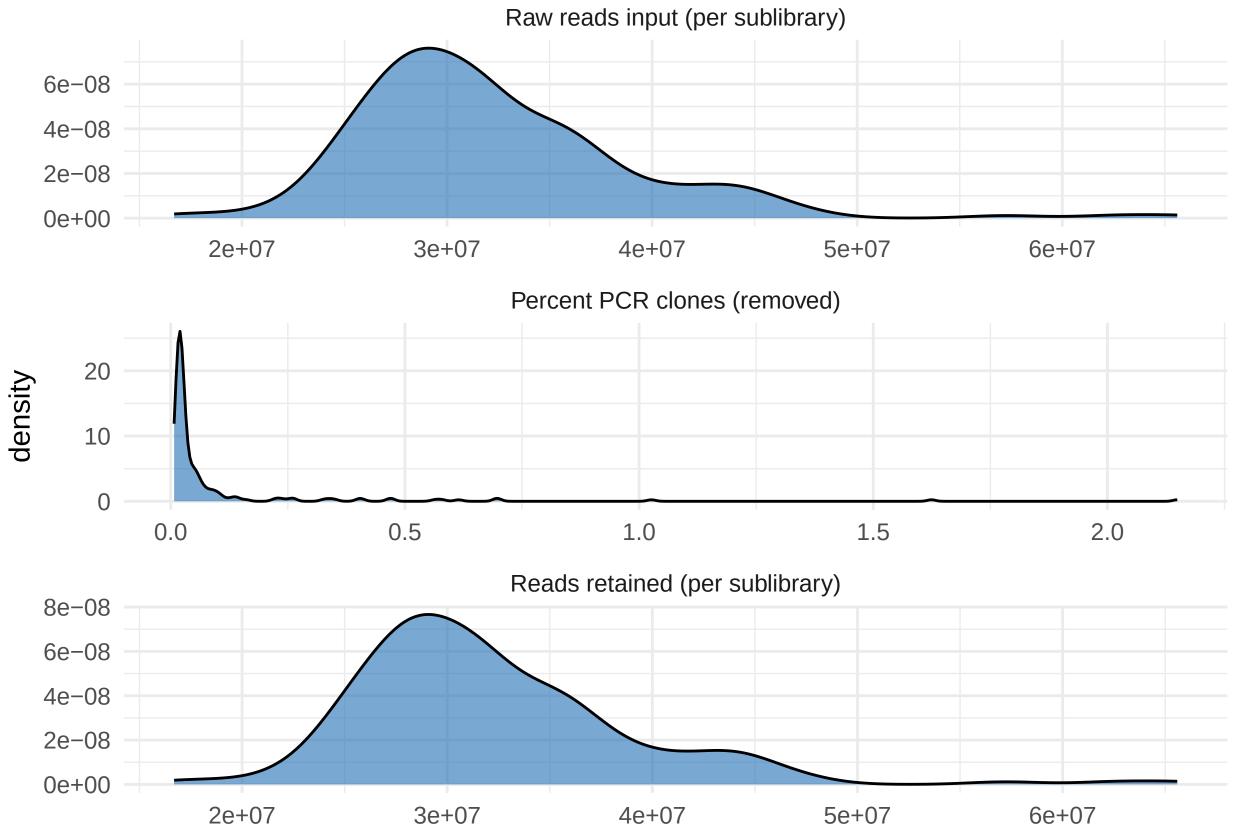


Figure S1: PCR clone removal statistics

```
ACAGTG  GTCCGC  DS.BA.PIK.U.5
GCCAAT  GTGAAA  DS.BA.DHI.U.1
CAGATC  GTGGCC  DS.BA.DHI.U.2
ACTTGA  GTTTCG  DS.BA.DHI.U.3
GATCAG  CGTACG  DS.BA.DHI.U.4
TAGCTT  GAGTGG  DS.BA.DHI.U.5
GGCTAC  ACTGAT  DS.BA.GA.U.1
CTTGTA  ATTCCT  DS.BA.GA.U.2
```

The 'process_radtags' command will demultiplex the data by separating out each sublibrary into the individual samples. It will then clean the data, and will remove low quality reads and discard reads where a barcode was not found.

3.4.2 Step 4b - Organize files

In a new directory, make sure the files are organized by species. In the `process_radtags` script, we specified that files be sent to `~/scratch/demux/*sublibrary_name*` (reasoning for this is in Step 4c), but files should manually be organized into species folders (i.e., `~/scratch/demux/*SPP*`) after `process_radtags` is performed. For example, the file "DS.MN.L01-DS.M.1.1.fq.gz" should be sent to the `~/scratch/demux/DS` directory.

Note: this is not automated at this point but it would be nice to automate the file moving process so it's not forgotten at this point.

3.4.3 Step 4c - Assess the raw, processed, and cleaned data

In the script for Step 4, we have specified that a new output folder be created for each sublibrary. The output folder is where all sample files and the log file will be dumped for each sublibrary. It is important to specify a different output folder if you have multiple sublibraries because we will be assessing the output log for each sublibrary individually (and otherwise, the log is overwritten when the script loops to a new sublibrary).

The utility `stacks-dist-extract` can be used to extract data from the log file. First, we examined the library-wide statistics to identify sublibraries where barcodes may have been misentered or where sequencing error may have occurred. We used:

```
stacks-dist-extract process_radtags.log total_raw_read_counts
```

to pull out data on the total number of sequences, the number of low-quality reads, whether barcodes were found or not, and the total number of retained reads per sublibary. Look over these to make sure there are no outliers or sublibraries that need to be checked and rerun.

Next, we used:

```
stacks-dist-extract process_radtags.log per_barcode_raw_read_counts
```

to analyze how well each sample performed. There are three important statistics to consider for each sample.

1. *The proportion of reads per sample for each sublibrary* indicates the proportion that each individual was processed and sequenced within the overall library. This is important to consider as cases where a single sample dominates the sublibrary may indicate contamination. (Field `prop_sample_per_library`).

2. *The number of reads retained for each sample* can be an indicator of coverage. It is most likely a good idea to remove samples with a very low number of reads. Where you decide to place the cutoff for low coverage samples is dependent on your dataset. For example, a threshold of 1 million reads is often used but this is not universal. (Field `retained_reads`).

3. *The proportion of reads retained for each sample* can also indicate low-quality samples and will give an idea of the variation in coverage across samples. (Field `prop_reads_retained_per_sample`).

Output for sublibraries for this step are summarized in process_radtags-library_output.csv.

Output for individual samples for this step are summarized in process_radtags-sample_output.csv.

The script 04c-process_radtags_stats.R was used to create many plots for easily assessing each statistic. Output from this step can be found in figures/process_radtags/ where figures are organized by species.

The script 04c-radtags_filter_summary.R summarizes the filtering results from all samples.

```
source("04_demux_filter/04c-radtags_filter_summary.R")
make_filterplot()
```

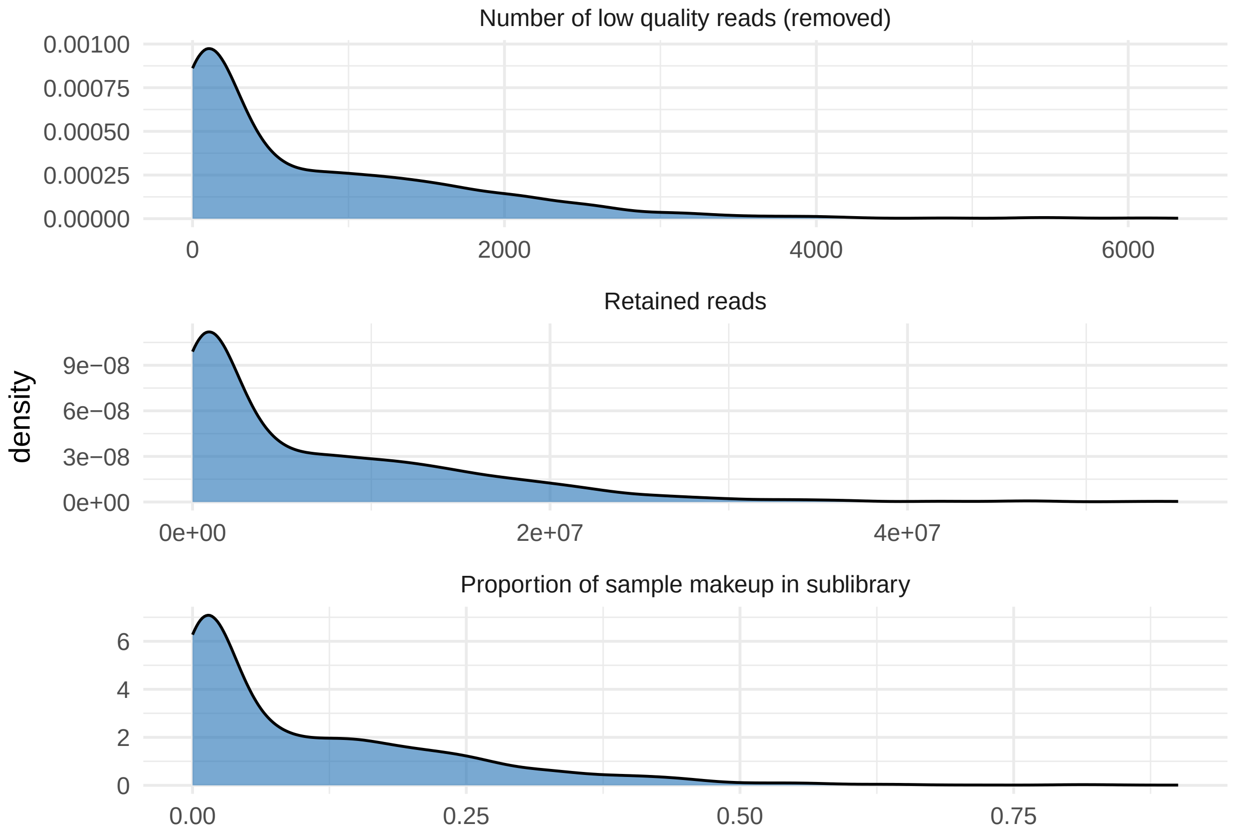


Figure S2: RAD tag processing statistics

3.4.4 Step 4d - Identify low-coverage and low-quality samples from

Using the same output log and the above statistics, we removed low-coverage and low-quality samples that may skew downstream analyses.

Samples were identified and removed via the following procedure:

1. First, samples that represented less than **1% of the sequenced sublibrary** were identified and removed. These samples correlate to low-read and low-coverage samples.
2. Next, a threshold of **1 million retained reads per sample** was used to remove any remaining low-read samples. Low-read samples correlate to low coverage and will lack enough raw reads to contribute to downstream analyses.

Good/kept samples are listed in process_radtags-kept_samples.csv.

Discarded samples are listed in process_radtags-discarded_samples.csv.

```
source("04_demux_filter/04c-radtags_filter_summary.R")
make_manual_discard_plot()
```

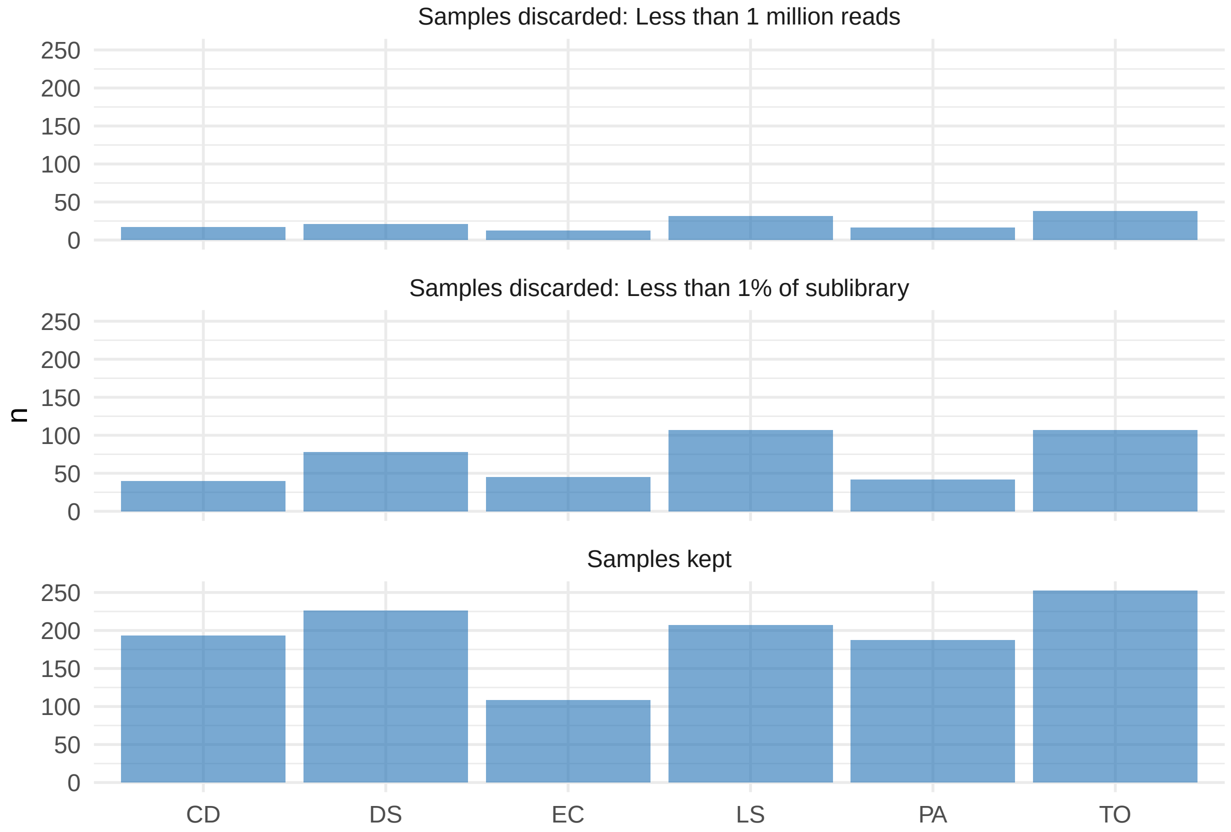


Figure S3: RAD tag manual filtering summary

Note: At this point, we started using Stacks 2.62 for its multi-threading capabilities. Functionality of the previous steps should be the same, however.

4 Generating Stacks Catalogs and Calling SNPs

4.1 Step 5 - Metapopulation Catalog Building and Parameter Search

Files can be found in the `05_ustacks_and_params/` directory.

Going forward, when we use the term **metapopulation**, we are referring to the collection of all samples within species among all cities where the species was present.

It is important to conduct preliminary analyses that will identify an optimal set of parameters for the dataset (see Step 5a). Following the parameter optimization, the program `ustacks` can be run to generate a catalog of loci.

4.1.1 Step 5a - Run `denovo_map.sh`

Stack assembly will differ based on several different aspects of the dataset(such as the study species, the RAD-seq method used, and/or the quality and quantity of DNA used). So it is important to use parameters that will maximize the amount of biological data obtained from stacks.

There are three main parameters to consider when doing this:

1. m = controls the minimum number of raw reads required to form a stack(implemented in `ustacks`)
2. M = controls the number of mismatches between stacks to to merge them into a putative locus (implemented in `ustacks`)
3. n = controls the number of mismatches allowed between stacks to merge into the catalog (implemented in `cstacks`)

There are two main ways to optimize parameterization:

1. an iterative method were you sequentially change each parameter while keeping the other parameters fixed (described in *Paris et al. 2017*), or
2. an iterative method were you sequentially change the values of M and n(keeping $M = n$) while fixing m = 3, and then test $m = 2$, 4 once the optimal $M = n$ is determined(described in *Rochette and Catchen 2017*, *Catchen 2020*).

We performed the second method and used the `denovo_map.sh` script to run the `denovo_map.pl` command to perform iterations. This script requires that we first choose a subset of samples to run the iterations on. The samples should be representative of the overall dataset; meaning they should include all populations and have similar read coverage numbers. Read coverage numbers can be assessed by looking at the descriptive statistics produced from Step 4c.

Place these samples in a text file (`popmap_test_samples.txt`) with the name of the sample and specify that all samples belong to the same population. For example, `popmap_test_samples.txt` should look like…

```
DS.BA.GA.U.1    A
DS.PX.BUF.M.5   A
DS.BO.HC4.M.1   A
...
```

It is important to have all representative samples treated as one population because you will assess outputs found across 80% of the individuals. The script will read this text file from the `--popmap` argument.

The script also requires that you specify an output directory after `-o`. This should be unique to the parameter you are testing… for example, if you are testing $M = 3$, then you could make a subdirectory labeled `stacks.M3` where all outputs from `denovo_map.sh` will be placed. Otherwise, for each iteration, the outputs will be overwritten and you will lose the log from the previous iteration. The `denovo_map.sh` script also requires that you direct it toward where your samples are stored, which is your directory built in Step 4b. Make sure to run the `--min-samples-per-pop 0.80` argument.

To decide which parameters to use, examine the following from each iteration:

1. the average sample coverage: This is obtained from the summary log in the `ustacks` section of `denovo_map.log`. If samples have a coverage <10x, you will have to rethink the parameters you use here.
2. the number of assembled loci shared by 80% of samples: This can be found in the `haplotypes.tsv` by counting the number of loci: `cat populations.haplotypes.tsv | grep -v ^"#" | wc -l`
3. the number of polymorphic loci shared by 80% of samples: This can be found in `populations.sumstats.tsv` or by counting `populations.hapstats.tsv`: `cat populations.hapstats.tsv | grep -v "^#" | wc -l`
4. the number of SNPs per locus shared by 80% of samples: found in `denovo_map.log` or by counting the number of SNPs in `populations.sumstats.tsv`: `populations.sumstats.tsv | grep -v ^"#" | wc -l`

The script `05a-param_opt-figures_script.R` was used to create plots for assessing the change in shared loci across parameter iterations.

Based on this optimization step, we used the following parameters:

Table S1: Final parameter optimization values for the Stacks pipeline.

Species	M (locus mismatches)	n (catalog mismatches)	m (minimum reads)
CD	8	8	3
DS	10	10	3
EC	8	8	3
LS	7	7	3
PA	5	5	3
TO	6	6	3

4.1.2 Step 5b - Run `ustacks`

`ustacks` builds *de novo* loci in each individual sample. We have designed the `ustacks` script so that the process requires three files:

- `05-ustacks_n.sh` : the shell script that executes `ustacks`
- `05-ustacks_id_n.txt` : the sample ID number
- `05-ustacks_samples_n.txt` : the sample names that correspond to the sample IDs

The sample ID should be derived from the `order_id` column(first column) on the master spreadsheet. It is unique (1-1736) across all of the samples.

The sample name is the corresponding name for each sample ID in the spreadsheet. E.g., sample ID "9" corresponds to sample name "DS.BA.DHI.U.4". Sample naming convention is species.city.site.management_type.replicate_plant.

`05-ustacks_n.sh` should have an out_directory (`-o` option) that will be used for all samples (e.g., `stacks/ustacks`). Files can be processed piecemeal into this directory. There should be three files for every sample in the output directory:

- `<samplename>.alleles.tsv.gz`
- `<samplename>.snps.tsv.gz`
- `<samplename>.tags.tsv.gz`

Multiple versions of the `05-ustacks_n.sh` script can be run in parallel (simply replace n in the three files above with the correct number).

A small number of samples (13) were discarded at this stage as the `ustacks` tool was unable to form any primary stacks corresponding to loci. See output/ustacks-discarded_samples.csv.

Table S2: Summary of samples discarded at the ustacks step of the Stacks pipeline.

ustacks discarded	Retained reads	Proportion of sub-library
no	10075192	0.1608982
yes	7490510	0.1230658

4.1.3 Step 5c - Correct File Names

This step contains a script `05b-fix_filenames.sh` which uses some simple regex to fix filenames that are output in previous steps. Stacks adds an extra "1" at some point at the end of the sample name which is not meaningful. The following files:

- DS.MN.L02-DS.M.3.1.alleles.tsv.gz
- DS.MN.L03-DS.U.2.1.tags.tsv.gz
- DS.MN.L09-DS.U.1.1.snps.tsv.gz

become:

- DS.MN.L02-DS.M.3.alleles.tsv.gz
- DS.MN.L03-DS.U.2.tags.tsv.gz
- DS.MN.L09-DS.U.1.snps.tsv.gz

The script currently gives some strange log output, so it can probably be optimized/improved. The script should be run from the directory where the changes need to be made. Files that have already been fixed will not be changed.

4.1.4 Step 5d - Choose catalog samples/files

In the next step, we will choose the files we want to go into the catalog. This involves a few steps:

1. Create a meaningful directory name. This could be the date (e.g., `stacks_22_01_25`).
2. Copy the `ustacks` output for all of the files you want to use in the reference from Step 5b. Remember this includes three files per sample. So if you have 20 samples you want to include in the reference catalog, you will transfer 3 x 20 = 60 files into the meaningful directory name. The three files per sample should follow this convention:

- `<samplename>.alleles.tsv.gz`
- `<samplename>.snps.tsv.gz`
- `<samplename>.tags.tsv.gz`

3. Remember the meaningful directory name. You will need it in Step 6.

4.2 Step 6 - Metapopulation catalog with `cstacks`

Files can be found in the `06_cstacks/` directory.

`cstacks` builds the locus catalog from all the samples specified. The accompanying script, `cstacks_SPECIES.sh` is relatively simple since it points to the directory containing all the sample files. It follows this format to point to that directory:

```
cstacks -P ~/directory ...
```

Make sure that you use the meaningful directory from Step 5c and that you have copied all the relevant files over. Otherwise this causes problems downstream. For example, you might edit the code to point to `~/scratch/stacks/stacks_22_01_25`.

```
cstacks -P ~/scratch/stacks/stacks_22_01_25 ...
```

The tricky thing is ensuring enough compute memory to run the entire process successfully. There is probably space to optimize this process.

The `cstacks` method uses a “population map” file, which in this project is `cstacks_popmap_SPECIES.txt`. This file specifies which samples to build the catalog from and categorizes them into your ‘populations’, or in this case, cities using two tab-delimited columns, e.g.:

```
DS.BA.GA.U.1    Baltimore
DS.BA.GA.U.2    Baltimore
DS.BA.GA.U.3    Baltimore
DS.BA.GA.U.4    Baltimore
DS.BA.GA.U.5    Baltimore
...
```

Make sure the samples in this file correspond to the input files located in e.g., `~/scratch/stacks/stacks_22_01_25`.

`cstacks` builds three files for use in all your samples (in this pipeline run), mirroring the sample files output by`ustacks`:

- `catalog.alleles.tsv.gz`
- `catalog.snps.tsv.gz`
- `catalog.tags.tsv.gz`

Table S3: Subset of samples used in SNP catalog creation.

Sample	Species	City
DS.BA.PIK.U.1	DS	BA
DS.BA.GA.U.4	DS	BA
DS.BA.LH-1.M.4	DS	BA
DS.BA.LH-3.M.1	DS	BA
DS.BA.WB.U.2	DS	BA
DS.BA.LL-4.M.5	DS	BA
DS.BA.LH-2.M.5	DS	BA
DS.BA.TRC.U.3	DS	BA
DS.BA.W3.M.2	DS	BA
DS.BA.RG-1.M.1	DS	BA
DS.BA.LL-3.M.3	DS	BA
DS.BA.RG-2.M.4	DS	BA
DS.BO.HC1.M.3	DS	BO
DS.BO.HC4.M.5	DS	BO
DS.BO.LC1.M.3	DS	BO
DS.BO.LC2.M.2	DS	BO
DS.BO.LC3.M.5	DS	BO
DS.BO.WL1.M.2	DS	BO
DS.BO.WL2.M.1	DS	BO
DS.BO.WL3.M.5	DS	BO
DS.BO.I4.U.1	DS	BO
DS.BO.R1.U.4	DS	BO
DS.BO.R2.U.2	DS	BO
DS.BO.R4.U.4	DS	BO
DS.MN.L05-DS.M.3	DS	MN
DS.MN.L09-DS.M.3	DS	MN
DS.MN.L11-DS.M.1	DS	MN
DS.MN.L02-DS.U.1	DS	MN
DS.MN.L02-DS.M.4	DS	MN
DS.MN.L03-DS.U.3	DS	MN
DS.MN.L04-DS.U.5	DS	MN
DS.MN.L06-DS.U.3	DS	MN
DS.MN.L07-DS.U.3	DS	MN
DS.MN.L09-DS.U.3	DS	MN
DS.MN.L11-DS.U.1	DS	MN
DS.MN.L11-DS.U.5	DS	MN
DS.PX.BUF.M.1	DS	PX
DS.PX.PIE.M.2	DS	PX
DS.PX.ALA.M.1	DS	PX
DS.PX.MTN.M.6	DS	PX
DS.PX.LAP.M.3	DS	PX
DS.PX.NUE.M.4	DS	PX
DS.PX.WES.M.2	DS	PX
DS.PX.DF1.M.1	DS	PX
DS.PX.ENC.M.1	DS	PX
DS.PX.DOW.M.1	DS	PX
DS.PX.DOW.M.4	DS	PX

Sample	Species	City
DS.PX.DF2.M.3	DS	PX
CD.BA.LA.U.2	CD	BA
CD.BA.TRC.U.3	CD	BA
CD.BA.WGP.M.2	CD	BA
CD.BA.LH-2.M.2	CD	BA
CD.BA.LL-4.M.1	CD	BA
CD.BA.PIK.U.2	CD	BA
CD.BA.WB.U.2	CD	BA
CD.BA.CP.U.4	CD	BA
CD.BA.FH.U.1	CD	BA
CD.BA.PSP.M.4	CD	BA
CD.BA.AA.U.4	CD	BA
CD.BA.RG-1.M.2	CD	BA
CD.BA.W3.M.3	CD	BA
CD.BA.GA.U.3	CD	BA
CD.BA.WBO.U.5	CD	BA
CD.LA.WHI.M.3	CD	LA
CD.LA.SEP.M.3	CD	LA
CD.LA.SEP.M.4	CD	LA
CD.LA.ROS.M.5	CD	LA
CD.LA.MR2.M.2	CD	LA
CD.LA.ALL.M.2	CD	LA
CD.LA.ALL.M.5	CD	LA
CD.LA.VAL.M.5	CD	LA
CD.LA.HAR.M.4	CD	LA
CD.LA.LUB.M.3	CD	LA
CD.LA.GLO.M.4	CD	LA
CD.LA.ZOO.M.3	CD	LA
CD.LA.NWH.M.5	CD	LA
CD.LA.KIN.M.3	CD	LA
CD.LA.KIN.M.5	CD	LA
CD.PX.CAM.U.5	CD	PX
CD.PX.MON.U.5	CD	PX
CD.PX.PKW.U.5	CD	PX
CD.PX.LAP.M.4	CD	PX
CD.PX.NES.U.4	CD	PX
CD.PX.PAL.M.3	CD	PX
CD.PX.ASU.M.1	CD	PX
CD.PX.NUE.M.5	CD	PX
CD.PX.WES.M.3	CD	PX
CD.PX.MAN.M.4	CD	PX
CD.PX.CLA.M.3	CD	PX
CD.PX.DF1.M.5	CD	PX
CD.PX.COY.M.5	CD	PX
CD.PX.RPC.M.3	CD	PX
CD.PX.ENC.M.2	CD	PX
EC.BA.LH-2.M.2	EC	BA
EC.BA.WBO.U.4	EC	BA
EC.BA.WB.U.5	EC	BA
EC.BA.FH.U.3	EC	BA
EC.BA.CP.U.2	EC	BA
EC.BA.TRC.U.3	EC	BA

Sample	Species	City
EC.BA.LL-4.M.4	EC	BA
EC.BA.WB.U.1	EC	BA
EC.BA.PIK.U.5	EC	BA
EC.BA.PSP.M.4	EC	BA
EC.BA.GA.U.2	EC	BA
EC.BA.LL-3.M.3	EC	BA
EC.BA.ML.U.1	EC	BA
EC.BA.TRC.U.5	EC	BA
EC.BA.ML.U.3	EC	BA
EC.LA.SGB.U.2	EC	LA
EC.LA.SGB.U.5	EC	LA
EC.LA.DUR.U.2	EC	LA
EC.LA.HOW.U.2	EC	LA
EC.LA.SAN.U.2	EC	LA
EC.LA.VER.U.1	EC	LA
EC.LA.VER.U.4	EC	LA
EC.LA.VB2.U.4	EC	LA
EC.LA.AC2.U.2	EC	LA
EC.LA.AC1.U.1	EC	LA
EC.LA.VB1.U.1	EC	LA
EC.LA.VB1.U.3	EC	LA
EC.LA.SGR.U.4	EC	LA
EC.LA.SGR.U.5	EC	LA
EC.LA.HOW.U.3	EC	LA
EC.PX.BUF.M.1	EC	PX
EC.PX.BUF.M.3	EC	PX
EC.PX.ALA.M.3	EC	PX
EC.PX.MTN.M.2	EC	PX
EC.PX.WES.M.1	EC	PX
EC.PX.WES.M.2	EC	PX
EC.PX.MAN.M.1	EC	PX
EC.PX.CLA.M.1	EC	PX
EC.PX.PSC.M.1	EC	PX
EC.PX.DF1.M.1	EC	PX
EC.PX.DOW.M.1	EC	PX
EC.PX.DOW.M.2	EC	PX
EC.PX.COY.M.2	EC	PX
EC.PX.COY.M.3	EC	PX
EC.PX.ALA.M.5	EC	PX
LS.BA.WB.U.1	LS	BA
LS.BA.WB.U.2	LS	BA
LS.BA.DHI.U.2	LS	BA
LS.BA.GA.U.1	LS	BA
LS.BA.PIK.U.3	LS	BA
LS.BA.PIK.U.5	LS	BA
LS.BA.CP.U.2	LS	BA
LS.BA.ML.U.2	LS	BA
LS.BA.WBO.U.3	LS	BA
LS.BO.WL3.M.4	LS	BO
LS.BO.I1.U.1	LS	BO
LS.BO.I2.U.1	LS	BO
LS.BO.WL2.M.2	LS	BO

Sample	Species	City
LS.BO.R1.U.2	LS	BO
LS.BO.R2.U.4	LS	BO
LS.BO.R3.U.3	LS	BO
LS.BO.HC4.M.3	LS	BO
LS.BO.LC4.M.2	LS	BO
LS.LA.VET.M.4	LS	LA
LS.LA.SSV.M.1	LS	LA
LS.LA.NAV.M.4	LS	LA
LS.LA.SHO.M.2	LS	LA
LS.LA.WES.M.3	LS	LA
LS.LA.GLO.M.3	LS	LA
LS.LA.HOW.U.5	LS	LA
LS.LA.SAN.U.2	LS	LA
LS.LA.ARR.U.2	LS	LA
LS.MN.L06-LS.U.2	LS	MN
LS.MN.L06-LS.U.5	LS	MN
LS.MN.L07-LS.U.4	LS	MN
LS.MN.L08-LS.U.5	LS	MN
LS.MN.L09-LS.U.3	LS	MN
LS.MN.L01-LS.M.4	LS	MN
LS.MN.L01-LS.U.3	LS	MN
LS.MN.L02-LS.U.1	LS	MN
LS.MN.L05-LS.U.2	LS	MN
LS.PX.MON.U.2	LS	PX
LS.PX.PKW.U.5	LS	PX
LS.PX.PIE.M.4	LS	PX
LS.PX.ALA.M.3	LS	PX
LS.PX.PAL.M.3	LS	PX
LS.PX.MAN.M.2	LS	PX
LS.PX.NUE.M.1	LS	PX
LS.PX.ENC.M.4	LS	PX
LS.PX.COY.M.3	LS	PX
PA.BA.PIK.U.1	PA	BA
PA.BA.LH-3.M.2	PA	BA
PA.BA.LH-3.M.3	PA	BA
PA.BA.WB.U.1	PA	BA
PA.BA.AA.U.1	PA	BA
PA.BA.WGP.M.3	PA	BA
PA.BA.LL-4.M.3	PA	BA
PA.BA.LA.U.2	PA	BA
PA.BA.LH-2.M.2	PA	BA
PA.BA.W3.M.3	PA	BA
PA.BA.RG-1.M.2	PA	BA
PA.BA.LL-3.M.5	PA	BA
PA.BO.I2.U.3	PA	BO
PA.BO.HC1.M.4	PA	BO
PA.BO.R3.U.2	PA	BO
PA.BO.HC4.M.5	PA	BO
PA.BO.R4.U.2	PA	BO
PA.BO.WL2.M.5	PA	BO
PA.BO.WL4.M.4	PA	BO
PA.BO.LC4.M.4	PA	BO

Sample	Species	City
PA.BO.HC2.M.1	PA	BO
PA.BO.R1.U.2	PA	BO
PA.BO.WL1.M.1	PA	BO
PA.BO.I1.U.5	PA	BO
PA.LA.ALL.M.5	PA	LA
PA.LA.SEP.M.1	PA	LA
PA.LA.SEP.M.5	PA	LA
PA.LA.WHI.M.2	PA	LA
PA.LA.ROS.M.5	PA	LA
PA.LA.LUB.M.2	PA	LA
PA.LA.GLO.M.2	PA	LA
PA.LA.ZOO.M.4	PA	LA
PA.LA.ZOO.M.5	PA	LA
PA.LA.NWH.M.2	PA	LA
PA.LA.KIN.M.4	PA	LA
PA.LA.POP.M.4	PA	LA
PA.PX.BUF.M.3	PA	PX
PA.PX.PIE.M.4	PA	PX
PA.PX.LAP.M.5	PA	PX
PA.PX.ALA.M.1	PA	PX
PA.PX.PAP.M.2	PA	PX
PA.PX.PAP.M.5	PA	PX
PA.PX.DF1.M.2	PA	PX
PA.PX.RPP.U.3	PA	PX
PA.PX.ENC.M.4	PA	PX
PA.PX.ENC.M.5	PA	PX
PA.PX.COY.M.1	PA	PX
PA.PX.BUF.M.2	PA	PX
TO.BA.WBO.U.4	TO	BA
TO.BA.CP.U.1	TO	BA
TO.BA.FH.U.1	TO	BA
TO.BA.LH-3.M.4	TO	BA
TO.BA.WGP.M.3	TO	BA
TO.BA.GA.U.4	TO	BA
TO.BA.PIK.U.4	TO	BA
TO.BA.PSP.M.1	TO	BA
TO.BA.RG-2.M.2	TO	BA
TO.BO.HC1.M.4	TO	BO
TO.BO.HC2.M.5	TO	BO
TO.BO.HC3.M.1	TO	BO
TO.BO.HC4.M.5	TO	BO
TO.BO.LC1.M.1	TO	BO
TO.BO.LC2.M.5	TO	BO
TO.BO.LC3.M.1	TO	BO
TO.BO.WL2.M.1	TO	BO
TO.BO.I2.U.3	TO	BO
TO.LA.WHI.M.5	TO	LA
TO.LA.HAR.M.4	TO	LA
TO.LA.MR1.M.1	TO	LA
TO.LA.GLO.M.5	TO	LA
TO.LA.ZOO.M.1	TO	LA
TO.LA.NWH.M.4	TO	LA

Sample	Species	City
TO.LA.VNS.M.2	TO	LA
TO.LA.PEP.M.5	TO	LA
TO.LA.COM.M.4	TO	LA
TO.MN.L11-TO.M.3	TO	MN
TO.MN.L02-TO.U.1	TO	MN
TO.MN.L04-TO.U.1	TO	MN
TO.MN.L06-TO.U.2	TO	MN
TO.MN.L08-TO.U.5	TO	MN
TO.MN.L09-TO.U.2	TO	MN
TO.MN.L11-TO.U.3	TO	MN
TO.MN.L05-TO.M.5	TO	MN
TO.MN.L08-TO.M.5	TO	MN
TO.PX.BUF.M.1	TO	PX
TO.PX.ALA.M.2	TO	PX
TO.PX.LAP.M.4	TO	PX
TO.PX.WES.M.1	TO	PX
TO.PX.CLA.M.1	TO	PX
TO.PX.DF1.M.1	TO	PX
TO.PX.DF2.M.1	TO	PX
TO.PX.COY.M.1	TO	PX
TO.PX.COY.M.6	TO	PX

4.3 Step 7 - Metapopulation locus matching with `sstacks`

Files can be found in the `07_sstacks/` directory.

All samples in the population (or all samples you want to include in the analysis) are matched against the catalog produced in `cstacks` with `sstacks`, run in script `stacks_SPECIES.sh` and `stacks_SPECIES_additional.sh`. It runs off of the samples based in the output directory *and* the listed samples in `sstacks_samples_SPECIES.txt` and `sstacks_samples_SPECIES_additional.txt` (respectively), so make sure all your files (sample and catalog, etc.) are there and match. `sstacks_samples_SPECIES.txt` takes the form:

```
DS.BA.GA.U.1
DS.BA.GA.U.2
DS.BA.GA.U.3
DS.BA.GA.U.4
DS.BA.GA.U.5
...
```

There should be a new file produced at this step for every sample in the output directory:

- `<samplename>.matches.tsv.gz`

A small number of samples generated very few matches to the catalog (such as only 4 loci matching, obviously not enough to draw any conclusions) and therefore aren't used in the next step. See output/sstacks-discarded_samples.csv.

4.4 Step 8 - Genotype probabilities with `polyRAD`

Files can be found in the `08_polyRAD/` directory.

4.4.1 Make `RADdata` object

We used the polyRAD package to call genotypes because many of our species are polyploid or have historical genome duplication. PolyRAD takes the catalog output (`catalog.alleles.tsv.gz`) and accompanying matches to the catalog (e.g., `CD.BA.AA.U.1.matches.tsv.gz`) to create genotype likelihoods for species with diploidy and/or polyploidy.

We used the catalog and match files to create a RADdata object class in R for each species. We ran this on the Rockfish compute cluster at Johns Hopkins University, with the `make_polyRAD_<spp>.R` script doing the brunt of the work. The R script was wrapped by `polyrad_make_<spp>.sh` to submit the script to the SLURM scheduler.

Relevant Parameters:

- `min.ind.with.reads` was set to 20% of samples. This means we discarded any loci not found in at least 20% of samples for each species.
- `min.ind.with.minor.allele` was set to 2. This means a locus must have at least this many samples with reads for the minor allele in order to be retained.

Requires:

- `popmap_<spp>_polyrad.txt`, a list of samples and population
- output from sstacks

Outputs:

- `<spp>_polyRADdata.rds`, RDS object (the RADdata object)

4.4.2 Calculate overdispersion

Next, we calculated overdispersion using the `polyRAD_overdispersion_<spp>.R` script, wrapped by `polyrad_overd_<spp>.sh` to submit the script to the SLURM scheduler.

Requires:

- `popmap_<spp>_polyrad.txt`, a list of samples and population
- `<spp>_polyRADdata.rds`, RDS object (the RADdata object) output from the previous step

Outputs:

- `<spp>_overdispersion.rds`, RDS object (the overdispersion test output)

4.4.3 Estimate genotypes

Next, we calculated filtered loci based on the expected Hind/He statistic and estimated population structure/genotypes using the `polyRAD_filter_<spp>.R` script, wrapped by `polyrad_filt_<spp>.sh` to submit the script to the SLURM scheduler.

We used the table in this tutorial, which estimated an inbreeding based on the ploidy, optimal overdispersion value, and mean Hind/He. These values are hardcoded in `polyRAD_filter_<spp>.R`.

Requires:

- `popmap_<spp>_polyrad.txt`, a list of samples and population
- `<spp>_polyRADdata.rds`, RDS object (the RADdata object) output from the previous step
- `<spp>_overdispersion.rds`, RDS object (the overdispersion test output) output from the previous step

Outputs:

- `<spp>_filtered_RADdata.rds`, RDS object (RADdata object filtered for appropriate Hind/He)
- `<spp>_IteratePopStructPCA.csv`, data output from the genotype estimate PCA, suitable for plotting
- `<spp>_estimatedgeno_RADdata.rds`, RDS object (RADdata object with genotype estimates)

4.4.4 Final filter and file cleanup

The output `<spp>_estimatedgeno_RADdata.rds` needs to be converted to genind and structure format for further analysis and steps. There is a little cleanup involved so the population information is retained. For example, Structure needs the population identity to be an integer, not a string. This set of functions can be run on a laptop.

At this stage, we also visually assessed the H_{ind}/H_e statistic versus the locus depth (see `check_coverage` inside the `convert_genomics.R` script). We removed the following samples from further analysis:

Table S4: Subset of samples discarded after genotype estimation using polyRAD.

Sample
CD.BA.PSP.M.1
CD.BA.DHI.U.2
CD.BA.DHI.U.3
CD.BA.RG-1.M.5
CD.BA.RG-1.M.4
DS.BO.WL1.M.4
DS.BO.I1.U.3
EC.BO.R4.U.1
LS.BO.HC2.M.5
LS.BO.LC4.M.3
LS.BO.R2.U.4
LS.BO.R2.U.1
PA.BA.LH-3.M.4
PA.BA.AA.U.3
PA.BA.AA.U.4
PA.PX.RPP.U.2
PA.BO.HC2.M.4
PA.PX.RPP.U.1
TO.BA.TRC.U.1
TO.BA.TRC.U.3
TO.BO.R4.U.1
TO.BA.TRC.U.2
TO.BO.R4.U.2
TO.BO.R2.U.2

```r
source("08_polyRAD/convert_genomics.R")
```

```r
convert_all()
```

4.5 Step 9 - Populations with `Structure`

Files, inlcuding model parameters, can be found in the `09_structure/` directory.

Structure documentation can be found here.

4.5.1 Running Structure

`polyRAD` outputs genotype probabilities in a format suitable for Structure. These files were named as:

```
CD_estimatedgeno.structure
DS_estimatedgeno.structure
```

```
EC_estimatedgeno.structure
LS_estimatedgeno.structure
PA_estimatedgeno.structure
TO_estimatedgeno.structure
```

We ran all species using a naive approach (not using prior information) with $K = 1, 2, 3, 4, 5$ (`MAXPOPS` argument). To search for the most appropriate K, We ran Structure through 5 replicate runs for each combination of species and K, with 10000 iterations discarded as burn-in and retained 20000 iterations. These runs created files that look like:

```
structure_out_CD1_naive_f        // K = 1, rep 1
structure_out_CD1_naive_rep2_f   // K = 1, rep 2
structure_out_CD1_naive_rep3_f   // K = 1, rep 3
structure_out_CD1_naive_rep4_f   // K = 1, rep 4
structure_out_CD1_naive_rep5_f   // K = 1, rep 5
structure_out_CD2_naive_f        // K = 2, rep 1
structure_out_CD2_naive_rep2_f   // K = 2, rep 2
structure_out_CD2_naive_rep3_f   // K = 2, rep 3
...
```

Within each species, we compressed the result files for all K and reps and submitted to Structure Harvester to choose the optimal K using the Delta-K method (see https://link.springer.com/article/10.1007/s12686-011-9548-7). Once the optimal K was selected per species, we re-ran Structure using a greater number of iterations (100000) for final output and plotting.

5 Analysis

5.1 NLCD Data

From the USGS:

> The U.S. Geological Survey (USGS), in partnership with several federal agencies, has developed and released four National Land Cover Database (NLCD) products over the past two decades: NLCD 1992, 2001, 2006, and 2011. This one is for data from 2016 and describes urban imperviousness.
>
> https://www.mrlc.gov/data/type/urban-imperviousness
>
> NLCD imperviousness products represent urban impervious surfaces as a percentage of developed surface over every 30-meter pixel in the United States. NLCD 2016 updates all previously released versions of impervious products for CONUS (NLCD 2001, NLCD 2006, NLCD 2011) along with a new date of impervious surface for 2016. New for NLCD 2016 is an impervious surface descriptor layer. This descriptor layer identifies types of roads, core urban areas, and energy production sites for each impervious pixel to allow deeper analysis of developed features.

First, we trimmed the large data. This makes a smaller `.rds` file for each city.

```r
source("R/trim_NLCD_spatial_data.R")
```

```r
create_spatial_rds_files()
```

5.2 Make maps of sampling locations

Next, we made plots for each city's sampling locations. Note that these only include sites that had viable SNPs.

```r
source("R/plot_map_of_samples.R")
```

```
make_all_urban_site_plots()
```

5.3 Continental population structure: population statistics by species

We used `polyrad::calcPopDiff()` to calculate continental population statistics for each species.

```
source("R/calc_continental_stats.R")
```

```
do_all_continental_stats()
```

```
# CD as an example
read.csv("output/population_stats/CD_continental_stats.csv")
```

```
##   X statistic      value
## 1 1     JostD 0.30579677
## 2 2       Gst 0.02735719
## 3 3       Fst 0.02812163
```

5.4 Continental population structure: Structure software results

Within each species, we compressed the result files for all K and reps and submitted to Structure Harvester to choose the optimal K using the Delta-K method (see https://link.springer.com/article/10.1007/s12686-011-9548-7).

The results were:

CD: K=3
DS: K=3
EC: K=2
LS: K=3
PA: K=4
TO: K=3

```
# This file contains output from various K from Structure..
read_csv("output/structure/structure_k_Pr.csv")
```

5.5 Continental population structure: Structure plots

The code below generates plots for Structure results.

```
source("R/plot_structure.R")
```

```
make_structure_multi_plot()
```

5.6 Validation of Structure results with sNMF

We ran sNMF as an alternative to Structure to validate the results. We coerced all polyploid data to diploid data to make the file types compatible with the sNMF function in R. The snmf() function computes an entropy criterion that evaluates the quality of fit of the statistical model to the data by using a cross-validation technique. We plotted the cross-entropy criterion for K=[2:10] for all species. Using the best K, we then selected the best of 10 runs in each K using the `which.min()` function.

```
source("R/sNMF.R")
```

The following runs sNMF and generates the figure. Note that in the pdf version of this document, the figure might appear on the next pages.

```r
do_all_sNMF()
```

5.7 Principal Components Analysis Plots

The following creates PCA plots from polyRAD data.

```r
source("R/plot_pca.R")
```

```r
plot_pcas()
```

In addition to coloring points by city, we also colored points by NLCD Urban Cover %. Note that in the pdf version of this document, the figure might appear on the next pages.

```r
plot_pcas_urbanness()
```

5.8 AMOVA

We performed hierarchical analysis of molecular variance (AMOVA; using GenoDive 3.06) based on the Rho-statistics, which is based on a Ploidy independent Infinite Allele Model. AMOVA is under the "Analysis" menu. Structure format files typically have only one level of population grouping. To ensure nestedness, we added a "City" level manually by going to "Data > Population grouping .." menu and adding "City". When running the AMOVA, we selected "Advanced" and selected Individual nested within Population nested within City. We used 999 permutations.

For bluegrass (PA), Minneapolis (MN) had a small sample size (2 individuals). We re-ran the AMOVA with these samples excluded but found similar results.

5.9 Local: F_{IS} - Homozygosity within population

We used GenoDive v. 3.0.6 to calculate F_{IS}. This gives a good estimate of whether there are more homozygotes than expected (positive number) or more heterozygotes than expected (negative number). Notably, GenoDive accommodates polyploids and reduces the bias on F_{IS} by performing a permutation test. By default, there are 999 permutations.

This can be run in GenoDive by selecting Analysis > Hardy-Weinberg > Heterozygosity-based (Nei) method.

```r
head(read.csv("output/population_stats/genodive_output_Fis.csv"))
```

```
##   Species Population  n   Fis
## 1      CD         BA 55 0.166
## 2      CD         LA 48 0.186
## 3      CD         PX 82 0.200
## 4      CD    Overall NA 0.187
## 5      DS         BA 55 0.208
## 6      DS         BO 52 0.252
```

5.10 Local: Genetic Diversity

We used GenoDive v. 3.0.6 to calculate several additional statistics.

This can be run in GenoDive by selecting Analysis > Genetic Diversity, and selecting "Calculate indices separately for every population" and selecting "Correct for unknown dosage of alleles" for the polyploid species.

- Num: Number of alleles
- Eff_num: Effective number of alleles
- Ho: Observed Heterozygosity
- Hs: Heterozygosity within populations

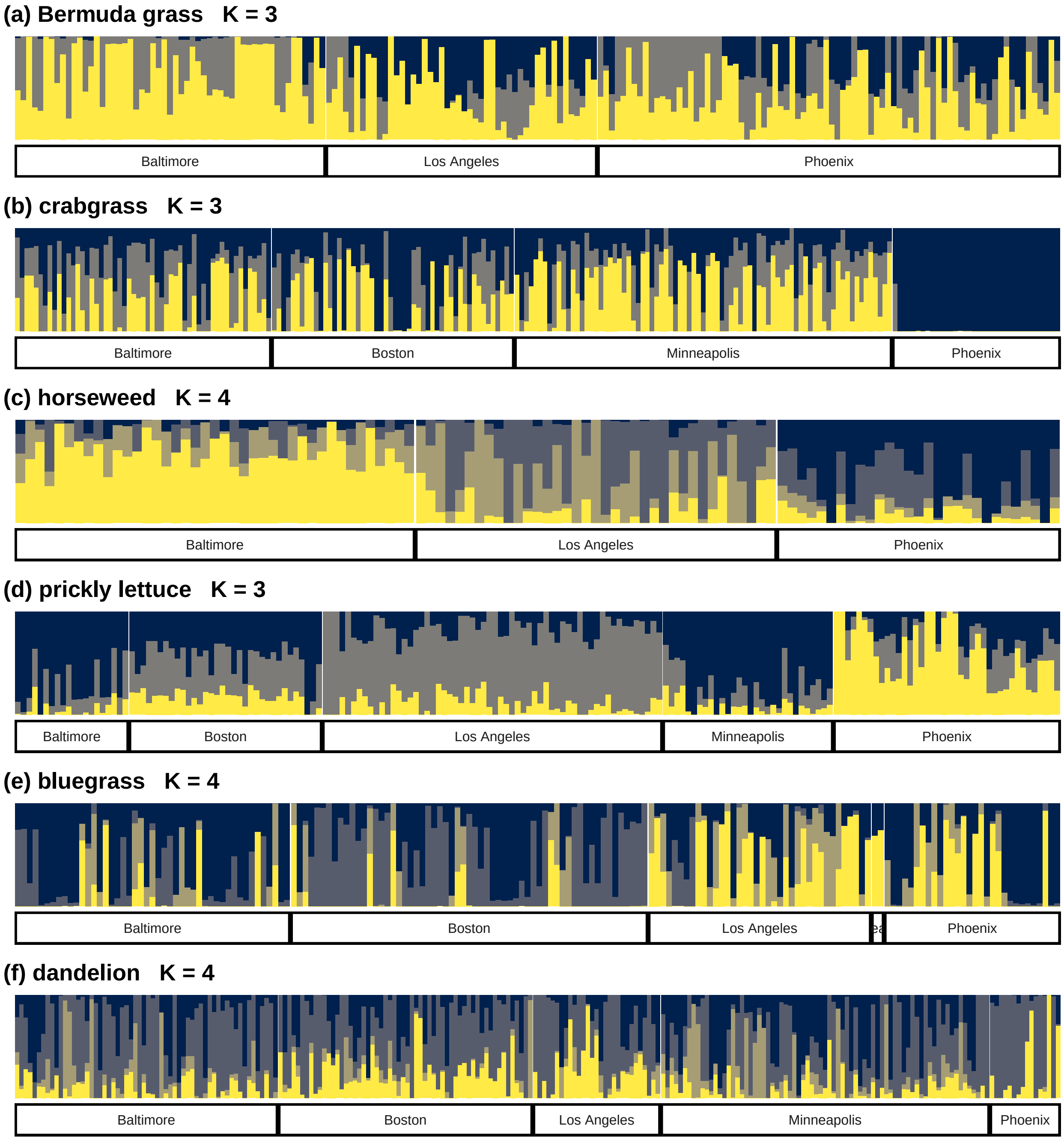


Figure S4: Ancestry coefficients obtained using `snmf()`. As with the Structure analysis, horseweed and prickly lettuce appear to have the most population structure. Phoenix crabgrass, horseweed, and prickly lettuce appear unique. In general, sNMF produced larger K for most species, which will create more sensitivity to admixture.

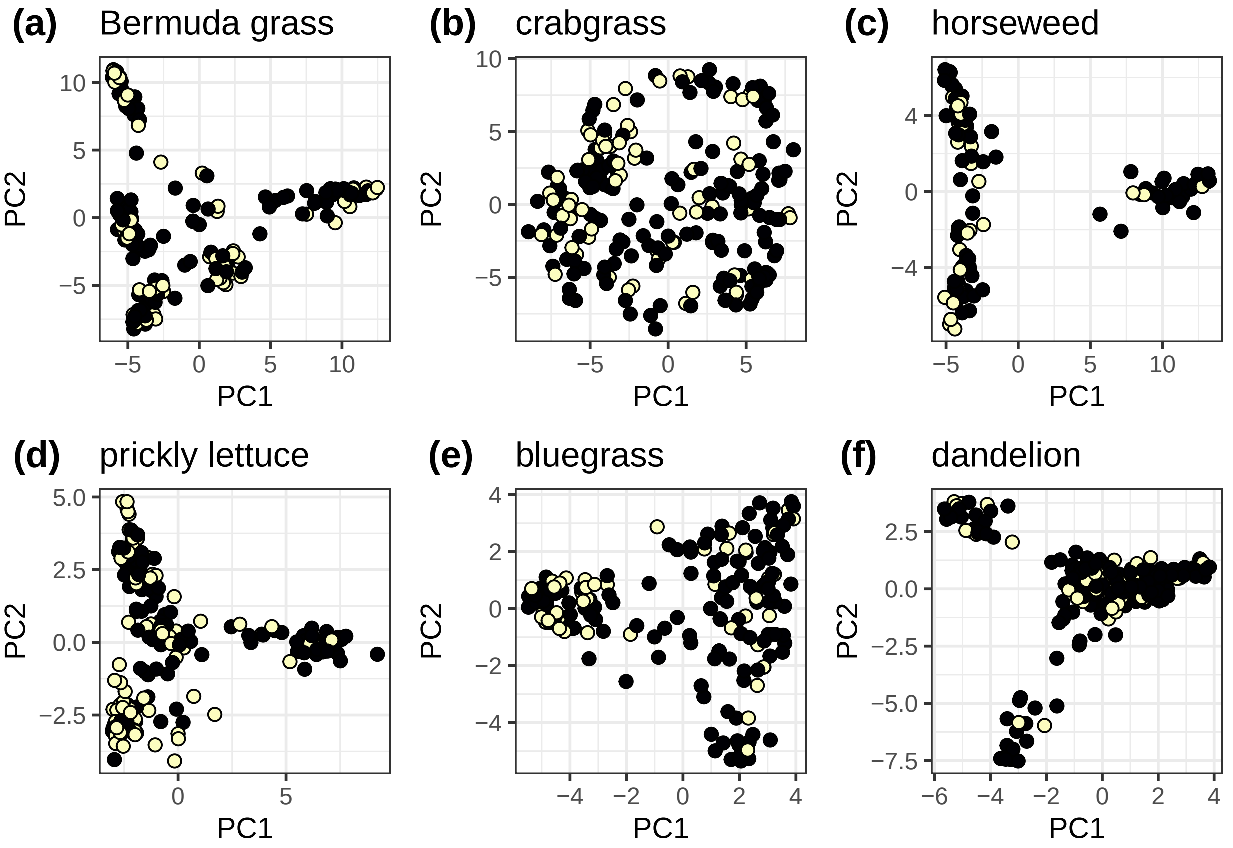


Figure S5: PCA colored by percent urban cover. Points in yellow are individuals from sites with >50% impervious cover. Black points are individuals from sites with <=50% impervious cover.

- Gis: Inbreeding coefficient

```
head(read.csv("output/population_stats/genodive_genetic_diversity.csv"))
```

```
##   spp city    Num Eff_num    Ho    Hs   Gis
## 1  CD   BA  8.242   4.530 0.627 0.748 0.163
## 2  CD   LA 12.664   6.470 0.683 0.827 0.174
## 3  CD   PX 14.192   6.978 0.672 0.830 0.191
## 4  DS   BA 10.611   5.069 0.637 0.768 0.171
## 5  DS   BO 10.588   5.230 0.612 0.778 0.214
## 6  DS   MN 11.696   5.233 0.623 0.771 0.192
```

5.11 Local: ρ - Pairwise comparison

We used GenoDive v. 3.0.6 to calculate pairise ρ (rho) among cities within species. Note that there is a p-value correction for testing multiple cities (species are treated as independent, however).

This can be run in GenoDive by selecting Pairwise Differentiation from the Analysis menu and selecting the "rho" statistic from the dropdown.

We used the following script to clean up the results.

```
source("R/rho.R")
compile_rho_table()
```

Table S5: Rho statistics for pairwise comparison between cities.

Species	City1	City2	rho	p-value	adjusted p-value
CD	PX	BA	0.050	0.001	**0.001**
CD	LA	BA	0.046	0.001	**0.001**
CD	PX	LA	0.015	0.001	**0.001**
DS	MN	BA	0.031	0.001	**0.0015**
DS	BO	BA	0.018	0.001	**0.0015**
DS	PX	BA	0.012	0.001	**0.0015**
DS	MN	BO	0.007	0.001	**0.0015**
DS	PX	BO	-0.002	0.875	0.955
DS	PX	MN	-0.002	0.955	0.955
EC	PX	BA	0.098	0.001	**0.001**
EC	PX	LA	0.087	0.001	**0.001**
EC	LA	BA	0.038	0.001	**0.001**
LS	PX	BA	0.077	0.001	**0.0011**
LS	PX	MN	0.069	0.001	**0.0011**
LS	PX	LA	0.061	0.001	**0.0011**
LS	PX	BO	0.056	0.001	**0.0011**
LS	MN	LA	0.039	0.001	**0.0011**
LS	LA	BA	0.038	0.001	**0.0011**
LS	BO	BA	0.032	0.001	**0.0011**
LS	MN	BO	0.021	0.001	**0.0011**
LS	LA	BO	0.010	0.001	**0.0011**
LS	MN	BA	0.009	0.002	**0.002**
PA	PX	BO	0.028	0.001	**0.0015**
PA	LA	BO	0.024	0.001	**0.0015**
PA	PX	BA	0.015	0.001	**0.0015**
PA	LA	BA	0.011	0.001	**0.0015**
PA	BO	BA	0.008	0.002	**0.0024**
PA	PX	LA	-0.002	0.972	0.972

Species	City1	City2	rho	p-value	adjusted p-value
TO	PX	BA	0.023	0.001	**0.0014**
TO	PX	MN	0.015	0.001	**0.0014**
TO	PX	BO	0.013	0.002	**0.0025**
TO	LA	BA	0.011	0.001	**0.0014**
TO	LA	BO	0.009	0.001	**0.0014**
TO	MN	LA	0.009	0.001	**0.0014**
TO	PX	LA	0.009	0.027	**0.03**
TO	BO	BA	0.008	0.001	**0.0014**
TO	MN	BO	0.008	0.001	**0.0014**
TO	MN	BA	0.001	0.098	0.098

5.12 Local: $\bar{r}_d$ - Linkage disequilibrium

We used `poppr::ia()` to calculate the standardized index of association of loci in the dataset ($\bar{r}_d$ or `rbarD`). We use the standardized index of association to avoid the influence of different sample sizes, as described by Agapow and Burt 2001.

When `p.rD` is small (<0.05) and rbarD is (relatively) higher, that is a sign that the population could be in linkage disequilibrium.

An interesting note from the documentation:

> It has been widely used as a tool to detect clonal reproduction within populations. Populations whose members are undergoing sexual reproduction, whether it be selfing or out-crossing, will produce gametes via meiosis, and thus have a chance to shuffle alleles in the next generation. Populations whose members are undergoing clonal reproduction, however, generally do so via mitosis. This means that the most likely mechanism for a change in genotype is via mutation. The rate of mutation varies from species to species, but it is rarely sufficiently high to approximate a random shuffling of alleles. The index of association is a calculation based on the ratio of the variance of the raw number of differences between individuals and the sum of those variances over each locus. You can also think of it as the observed variance over the expected variance.

There is a nice description here.

```r
source("R/rbarD.R")
calc_rbarD()
```

```r
head(read.csv("output/population_stats/rbarD.csv"))
```

```
##   spp city  n       Ia  p.Ia     rbarD  p.rD
## 1  CD   BA 55 664.7655 0.001 0.2950534 0.001
## 2  CD   LA 48 470.5913 0.001 0.2070064 0.001
## 3  CD   PX 82 634.5787 0.001 0.2792566 0.001
## 4  DS   BA 55 557.7334 0.001 0.2123881 0.001
## 5  DS   BO 52 896.0906 0.001 0.3398873 0.001
## 6  DS   MN 81 578.2494 0.001 0.2192197 0.001
```

5.13 Isolation by distance and environment

We used multiple matrix regression with randomization (MMRR) to determine the relative contributions of isolation by distance (i.e., an association between genetic and geographic distances) and isolation by environment (i.e., an association between genetic and environmental distances).

5.13.1 Genetic distance

We calculated the genetic dissimilarity matrix using the Cavalli-Sforza (Chord) distance metric in GenoDive. While this can also be done using `adegenet`, we don't want to make assumptions about ploidy.

5.13.2 Geographic distance

We took the traditional approach to creating a geographic dissimilarity matrix (based on latitude and longitude) using euclidean distance.

5.13.3 Environmental data and distance

Environmental variables include the monthly averages in the middle of the day for:

- air temperature at 5cm above ground
- air temperature at 1.2m above ground
- soil temperature at 2.5cm below ground
- RH (relative humidity) at 5cm above ground
- RH at 1.2m above ground

Variables were extracted from historic datasets and modeled using a microclimate model. More information can be found on the NicheMapR website (how the model works, what variables can be manipulated and what you can model, vignettes for running models in R).

This method was chosen because it takes data from global datasets (you can use both historic and current or pick specific years) but then accounts for site-specific variables (we can change the % shade, the slope or aspect of the landscape, and it considers elevation, average cloud cover, etc.). Here's the list of all the different models/datasets we're able to can pull from. It's meant for mechanistic niche modeling.

Variables in the file `site_data_DUC_environvars.csv` are all for the monthly averages at noon (12pm - hottest part of the day!) and are extreme. In other words, they are maximums.

Note that this Stack Overflow post is helpful with installing `NicheMapR`.

```r
# devtools::install_github('mrke/NicheMapR') library(NicheMapR)
# test_site_coords <- c(sites[1,]$lat, sites[1,]$long)
# test_distance_to_city_center_km <-
# sites[1,]$distance_to_city_center_km micros_ <- micro_usa(loc =
# test_site_coords) loc <- c(-89.40, 43.07) micro <- micro_global(loc =
# loc)
```

Genetic distance was generated the same way as geographic distance above (euclidean distance). These are the four environmental variables mentioned in the main manuscript, although more environmental variables are present in the raw data: % Urban cover, Distance to city center, April soil temperature, and July soil temperature. In the raw data these appear as `nlcd_urban_pct`, `distance_to_city_center`, `soiltemp_2.5cm_Apr_12pm`, `soiltemp_2.5cm_Jul_12pm`.

5.13.4 Analysis

Note that for this analysis, we treated each *sampling site* as a distinct location. There would not be enough power to do a distance matrix among 3-5 cities. Code for generating matrices, running the MMRR, and generating figures can be found in the source code below.

```r
source("R/IBD_IBE_MMRR.R")
```

```r
make_mmrr_plot()
```

Below are the results of the MMRR, with all cities in the same model. Species are treated as independent, separate models.

Table S6: Statistics from running 9999 permutations ('Reps') via MMRR.

var	estimate	p	95% Lower	95% Upper	spp
distance_to_city_center	0.2200000	**0.0119**	0.1778817	0.2717601	CD
geodist	0.2800000	0.2142	0.1049370	0.4568245	CD
Intercept	0.0000000	0.2807	-0.0458878	0.0458878	CD
nlcd_urban_pct	-0.0400000	0.5666	-0.0811225	0.0109601	CD
soiltemp_Apr	-0.6900000	0.0598	-1.0028857	-0.3770496	CD
soiltemp_Jul	0.5900000	**0.0027**	0.3993291	0.7718615	CD
R-Squared:	0.0900707	NA	NA	NA	CD
F-Statistic:	23.1628458	NA	NA	NA	CD
F p-value:	0.0035000	NA	NA	NA	CD
distance_to_city_center	-0.0700000	0.4747	-0.1006672	-0.0408287	DS
geodist	0.2600000	**1e-04**	0.2139693	0.3096376	DS
Intercept	0.0000000	0.0803	-0.0287769	0.0287769	DS
nlcd_urban_pct	0.0300000	0.6047	-0.0002724	0.0574211	DS
soiltemp_Apr	1.1000000	**1e-04**	0.9181513	1.2839537	DS
soiltemp_Jul	-1.7700000	**1e-04**	-1.9569820	-1.5730522	DS
R-Squared:	0.2840368	NA	NA	NA	DS
F-Statistic:	185.6648629	NA	NA	NA	DS
F p-value:	0.0001000	NA	NA	NA	DS
distance_to_city_center	0.2800000	**0.0013**	0.2228914	0.3439358	EC
geodist	-0.0400000	0.6365	-0.1553802	0.0721141	EC
Intercept	0.0000000	0.4512	-0.0572346	0.0572346	EC
nlcd_urban_pct	-0.0500000	0.4057	-0.1136514	0.0057864	EC
soiltemp_Apr	-0.1900000	**0.0474**	-0.3897004	0.0088973	EC
soiltemp_Jul	0.6500000	**1e-04**	0.5058642	0.7950112	EC
R-Squared:	0.3290669	NA	NA	NA	EC
F-Statistic:	54.4412294	NA	NA	NA	EC
F p-value:	0.0001000	NA	NA	NA	EC
distance_to_city_center	-0.3600000	**5e-04**	-0.3978213	-0.3280028	LS
geodist	-0.1300000	0.0945	-0.1782062	-0.0759298	LS
Intercept	0.0000000	0.8986	-0.0336011	0.0336011	LS
nlcd_urban_pct	0.0400000	0.4429	0.0045620	0.0731658	LS
soiltemp_Apr	0.2000000	0.1098	0.1140996	0.2955602	LS
soiltemp_Jul	0.0300000	0.8262	-0.0412286	0.1094663	LS
R-Squared:	0.1615783	NA	NA	NA	LS
F-Statistic:	77.4723329	NA	NA	NA	LS
F p-value:	0.0002000	NA	NA	NA	LS
distance_to_city_center	-0.1300000	0.2887	-0.1754371	-0.0821790	PA
geodist	0.4600000	**0.0073**	0.3252127	0.6019750	PA
Intercept	0.0000000	0.5196	-0.0464543	0.0464543	PA
nlcd_urban_pct	0.0000000	0.9576	-0.0496728	0.0433418	PA
soiltemp_Apr	-0.4200000	0.1529	-0.6864671	-0.1551349	PA
soiltemp_Jul	0.1500000	0.5495	-0.0239224	0.3235452	PA
R-Squared:	0.0674673	NA	NA	NA	PA
F-Statistic:	16.9295489	NA	NA	NA	PA
F p-value:	0.0976000	NA	NA	NA	PA
distance_to_city_center	0.0500000	0.5852	0.0215944	0.0826828	TO
geodist	-0.0600000	0.3368	-0.1008273	-0.0113832	TO
Intercept	0.0000000	0.4746	-0.0301865	0.0301865	TO
nlcd_urban_pct	0.0000000	0.9869	-0.0315460	0.0294925	TO
soiltemp_Apr	0.3200000	0.0779	0.2232558	0.4132483	TO

var	estimate	p	95% Lower	95% Upper	spp
soiltemp_Jul	-0.4400000	**0.0465**	-0.5199541	-0.3523079	TO
R-Squared:	0.0424340	NA	NA	NA	TO
F-Statistic:	25.2060335	NA	NA	NA	TO
F p-value:	0.1653000	NA	NA	NA	TO

We also repeated this within city.

Table S7: Overall model p-values from MMRR, subset by each city. Adjusted p-values are corrected using the Benjamini and Hochberg method within each species' results.

spp	city	R-Squared:	F-Statistic:	F p-value:	p adjusted
CD	BA	0.1625629	3.8435668	0.2290	0.3435
CD	LA	0.0945152	1.2525691	0.6516	0.6516
CD	PX	0.2804009	17.5348220	0.0004	**0.0012**
DS	BA	0.1332715	3.9978609	0.2716	0.6264
DS	BO	0.0919259	2.6320243	0.5485	0.721
DS	MN	0.1161402	6.4912166	0.3132	0.6264
DS	PX	0.0910634	1.2022407	0.7210	0.721
EC	BA	0.0629354	0.9671366	0.9130	0.913
EC	LA	0.1322816	1.1890918	0.5913	0.887
EC	PX	0.3210133	4.6332725	0.0534	0.1602
LS	BA	0.6460388	8.0307411	0.0998	0.1663
LS	BO	0.0700753	1.2810507	0.8163	0.8468
LS	LA	0.6583433	56.6512998	0.0189	0.0945
LS	MN	0.2990934	5.1206829	0.0729	0.1663
LS	PX	0.0540128	0.6851617	0.8468	0.8468
PA	BA	0.2981997	5.0988812	0.0943	0.3772
PA	BO	0.1112364	3.2541242	0.4281	0.4669
PA	LA	0.2081501	2.0503515	0.3566	0.4669
PA	PX	0.2515098	2.0161370	0.4669	0.4669
TO	BA	0.0283788	0.8587052	0.9491	0.9491
TO	BO	0.0802685	2.2691187	0.5396	0.6745
TO	LA	0.6014791	18.1113470	0.0378	0.0945
TO	MN	0.0643590	2.8064698	0.4602	0.6745
TO	PX	0.6599428	8.5389990	0.0319	0.0945

Table S8: Parameter estimates and statistics from running 9999 permutations ('Reps') via MMRR, subset by each city. Note that p-values are not adjusted for multiple testing.

var	estimate	p	95% Lower	95% Upper	spp	city
distance_to_city_center	0.0200000	0.8973	-0.1523737	0.1905424	CD	BA
geodist	-0.3500000	**0.0211**	-0.5185338	-0.1879233	CD	BA
Intercept	0.0000000	0.4032	-0.1519816	0.1519816	CD	BA
nlcd_urban_pct	-0.0600000	0.6558	-0.2148227	0.1005574	CD	BA
soiltemp_Apr	-0.2800000	0.32	-0.5626588	-0.0028176	CD	BA
soiltemp_Jul	0.3100000	0.3282	0.0271258	0.5893149	CD	BA
R-Squared:	0.1625629	NA	NA	NA	CD	BA
F-Statistic:	3.8435668	NA	NA	NA	CD	BA

var	estimate	p	95% Lower	95% Upper	spp	city
F p-value:	0.2290000	NA	NA	NA	CD	BA
distance_to_city_center	0.1400000	0.4463	-0.0985715	0.3778403	CD	LA
geodist	-0.3100000	0.2809	-0.5930246	-0.0204906	CD	LA
Intercept	0.0000000	0.0997	-0.2036737	0.2036737	CD	LA
nlcd_urban_pct	0.2400000	0.1315	0.0298573	0.4468668	CD	LA
soiltemp_Apr	0.1800000	0.4165	-0.1013289	0.4517609	CD	LA
soiltemp_Jul	0.0800000	0.6539	-0.1303464	0.2993074	CD	LA
R-Squared:	0.0945152	NA	NA	NA	CD	LA
F-Statistic:	1.2525691	NA	NA	NA	CD	LA
F p-value:	0.6516000	NA	NA	NA	CD	LA
distance_to_city_center	0.3400000	**0.0373**	0.2072518	0.4807748	CD	PX
geodist	0.4000000	**0.0022**	0.2770121	0.5152218	CD	PX
Intercept	0.0000000	0.8503	-0.0932033	0.0932033	CD	PX
nlcd_urban_pct	-0.0300000	0.7013	-0.1322619	0.0657577	CD	PX
soiltemp_Apr	0.0000000	0.9831	-0.2153432	0.2071494	CD	PX
soiltemp_Jul	-0.2500000	0.0577	-0.4516839	-0.0537647	CD	PX
R-Squared:	0.2804009	NA	NA	NA	CD	PX
F-Statistic:	17.5348220	NA	NA	NA	CD	PX
F p-value:	0.0004000	NA	NA	NA	CD	PX
distance_to_city_center	-0.4400000	**0.0109**	-0.6104353	-0.2790133	DS	BA
geodist	0.2500000	**0.0428**	0.0861417	0.4221049	DS	BA
Intercept	0.0000000	**0.0113**	-0.1347723	0.1347723	DS	BA
nlcd_urban_pct	-0.0100000	0.9143	-0.1487627	0.1231785	DS	BA
soiltemp_Apr	0.0800000	0.802	-0.1839206	0.3359334	DS	BA
soiltemp_Jul	-0.0400000	0.8914	-0.2958051	0.2163047	DS	BA
R-Squared:	0.1332715	NA	NA	NA	DS	BA
F-Statistic:	3.9978609	NA	NA	NA	DS	BA
F p-value:	0.2716000	NA	NA	NA	DS	BA
distance_to_city_center	-0.5700000	0.103	-0.9098795	-0.2260767	DS	BO
geodist	0.3600000	0.1553	0.0217352	0.6981460	DS	BO
Intercept	0.0000000	0.7256	-0.1379493	0.1379493	DS	BO
nlcd_urban_pct	-0.0700000	0.6664	-0.2170233	0.0766400	DS	BO
soiltemp_Apr	0.0100000	0.9879	-0.2449152	0.2551026	DS	BO
soiltemp_Jul	-0.0600000	0.8223	-0.3112849	0.1870680	DS	BO
R-Squared:	0.0919259	NA	NA	NA	DS	BO
F-Statistic:	2.6320243	NA	NA	NA	DS	BO
F p-value:	0.5485000	NA	NA	NA	DS	BO
distance_to_city_center	0.0500000	0.7321	-0.0903497	0.1911004	DS	MN
geodist	0.1200000	0.4199	-0.0192034	0.2590773	DS	MN
Intercept	0.0000000	0.5525	-0.0985695	0.0985695	DS	MN
nlcd_urban_pct	-0.0100000	0.9417	-0.1141939	0.0980041	DS	MN
soiltemp_Apr	0.3200000	0.22	0.1617489	0.4864167	DS	MN
soiltemp_Jul	-0.0500000	0.8532	-0.2128943	0.1185175	DS	MN
R-Squared:	0.1161402	NA	NA	NA	DS	MN
F-Statistic:	6.4912166	NA	NA	NA	DS	MN
F p-value:	0.3132000	NA	NA	NA	DS	MN
distance_to_city_center	0.0000000	0.9967	-0.3284950	0.3267356	DS	PX
geodist	-0.2400000	0.3248	-0.4784123	-0.0028135	DS	PX
Intercept	0.0000000	0.7725	-0.2040615	0.2040615	DS	PX
nlcd_urban_pct	0.1000000	0.5874	-0.1210604	0.3129019	DS	PX
soiltemp_Apr	-0.2400000	0.4982	-0.5858385	0.1035301	DS	PX
soiltemp_Jul	0.0700000	0.7285	-0.2302025	0.3678408	DS	PX

var	estimate	p	95% Lower	95% Upper	spp	city
R-Squared:	0.0910634	NA	NA	NA	DS	PX
F-Statistic:	1.2022407	NA	NA	NA	DS	PX
F p-value:	0.7210000	NA	NA	NA	DS	PX
distance_to_city_center	0.0900000	0.6916	-0.1402286	0.3268280	EC	BA
geodist	0.0400000	0.8507	-0.1945208	0.2703859	EC	BA
Intercept	0.0000000	0.6794	-0.1888722	0.1888722	EC	BA
nlcd_urban_pct	-0.0900000	0.631	-0.2905650	0.1030852	EC	BA
soiltemp_Apr	-0.0700000	0.8783	-0.4155600	0.2790409	EC	BA
soiltemp_Jul	0.2600000	0.4575	-0.0838199	0.6081152	EC	BA
R-Squared:	0.0629354	NA	NA	NA	EC	BA
F-Statistic:	0.9671366	NA	NA	NA	EC	BA
F p-value:	0.9130000	NA	NA	NA	EC	BA
distance_to_city_center	-0.9400000	0.0832	-1.6122099	-0.2598591	EC	LA
geodist	0.0800000	0.7098	-0.2349552	0.3927614	EC	LA
Intercept	0.0000000	0.6185	-0.2485106	0.2485106	EC	LA
nlcd_urban_pct	0.5600000	0.1261	0.0411702	1.0872646	EC	LA
soiltemp_Apr	0.0100000	0.9704	-0.2593919	0.2765288	EC	LA
soiltemp_Jul	0.4700000	0.1918	0.0759790	0.8600901	EC	LA
R-Squared:	0.1322816	NA	NA	NA	EC	LA
F-Statistic:	1.1890918	NA	NA	NA	EC	LA
F p-value:	0.5913000	NA	NA	NA	EC	LA
distance_to_city_center	0.3200000	0.1977	0.0521247	0.5853019	EC	PX
geodist	0.2700000	0.144	0.0401394	0.4922747	EC	PX
Intercept	0.0000000	0.7511	-0.1955532	0.1955532	EC	PX
nlcd_urban_pct	0.2100000	0.2363	0.0068546	0.4064400	EC	PX
soiltemp_Apr	0.0800000	0.8098	-0.3348882	0.4934040	EC	PX
soiltemp_Jul	-0.0400000	0.8632	-0.4295061	0.3531004	EC	PX
R-Squared:	0.3210133	NA	NA	NA	EC	PX
F-Statistic:	4.6332725	NA	NA	NA	EC	PX
F p-value:	0.0534000	NA	NA	NA	EC	PX
distance_to_city_center	1.0700000	0.0922	0.6844511	1.4647814	LS	BA
geodist	-1.3000000	**0.0261**	-1.6916882	-0.9155998	LS	BA
Intercept	0.0000000	0.527	-0.2138829	0.2138829	LS	BA
nlcd_urban_pct	0.1200000	0.5297	-0.1059047	0.3510392	LS	BA
soiltemp_Apr	0.3100000	0.4468	-0.1074624	0.7217407	LS	BA
soiltemp_Jul	-0.1500000	0.6985	-0.5662459	0.2640221	LS	BA
R-Squared:	0.6460388	NA	NA	NA	LS	BA
F-Statistic:	8.0307411	NA	NA	NA	LS	BA
F p-value:	0.0998000	NA	NA	NA	LS	BA
distance_to_city_center	-0.0200000	0.979	-0.5452077	0.5105561	LS	BO
geodist	-0.1100000	0.7865	-0.6290220	0.4087116	LS	BO
Intercept	0.0000000	0.4855	-0.1729823	0.1729823	LS	BO
nlcd_urban_pct	-0.0100000	0.9706	-0.1862877	0.1713059	LS	BO
soiltemp_Apr	-0.2500000	0.548	-0.6047174	0.1044697	LS	BO
soiltemp_Jul	0.0700000	0.8509	-0.2791814	0.4131979	LS	BO
R-Squared:	0.0700753	NA	NA	NA	LS	BO
F-Statistic:	1.2810507	NA	NA	NA	LS	BO
F p-value:	0.8163000	NA	NA	NA	LS	BO
distance_to_city_center	-0.4000000	**0.0266**	-0.5467220	-0.2455640	LS	LA
geodist	-0.4700000	0.0581	-0.6242402	-0.3251064	LS	LA
Intercept	0.0000000	0.9882	-0.0795399	0.0795399	LS	LA
nlcd_urban_pct	0.0600000	0.5033	-0.0277185	0.1524868	LS	LA

var	estimate	p	95% Lower	95% Upper	spp	city
soiltemp_Apr	0.3100000	0.0903	0.2169813	0.3974631	LS	LA
soiltemp_Jul	-0.0200000	0.8378	-0.1085275	0.0610078	LS	LA
R-Squared:	0.6583433	NA	NA	NA	LS	LA
F-Statistic:	56.6512998	NA	NA	NA	LS	LA
F p-value:	0.0189000	NA	NA	NA	LS	LA
distance_to_city_center	0.0200000	0.9409	-0.1737961	0.2059559	LS	MN
geodist	-0.4700000	**0.0094**	-0.6646514	-0.2744961	LS	MN
Intercept	0.0000000	0.5669	-0.1791944	0.1791944	LS	MN
nlcd_urban_pct	-0.1200000	0.4841	-0.3106588	0.0783110	LS	MN
soiltemp_Apr	0.3300000	0.3703	0.0181070	0.6319196	LS	MN
soiltemp_Jul	-0.4100000	0.197	-0.7183980	-0.0980235	LS	MN
R-Squared:	0.2990934	NA	NA	NA	LS	MN
F-Statistic:	5.1206829	NA	NA	NA	LS	MN
F p-value:	0.0729000	NA	NA	NA	LS	MN
distance_to_city_center	-0.1400000	0.5914	-0.4206643	0.1498583	LS	PX
geodist	0.0600000	0.8202	-0.2903303	0.4072557	LS	PX
Intercept	0.0000000	0.8636	-0.2081790	0.2081790	LS	PX
nlcd_urban_pct	-0.1200000	0.3594	-0.3319855	0.0912733	LS	PX
soiltemp_Apr	-0.1600000	0.8467	-1.0258576	0.7006935	LS	PX
soiltemp_Jul	0.3500000	0.6416	-0.3921604	1.0983270	LS	PX
R-Squared:	0.0540128	NA	NA	NA	LS	PX
F-Statistic:	0.6851617	NA	NA	NA	LS	PX
F p-value:	0.8468000	NA	NA	NA	LS	PX
distance_to_city_center	0.5800000	**0.0052**	0.3864547	0.7807829	PA	BA
geodist	-0.1300000	0.3044	-0.3352591	0.0832338	PA	BA
Intercept	0.0000000	0.28	-0.1793086	0.1793086	PA	BA
nlcd_urban_pct	-0.1500000	0.4048	-0.3395375	0.0414073	PA	BA
soiltemp_Apr	-0.1200000	0.4554	-0.3888812	0.1476453	PA	BA
soiltemp_Jul	0.1000000	0.7842	-0.1737255	0.3733119	PA	BA
R-Squared:	0.2981997	NA	NA	NA	PA	BA
F-Statistic:	5.0988812	NA	NA	NA	PA	BA
F p-value:	0.0943000	NA	NA	NA	PA	BA
distance_to_city_center	0.0900000	0.8053	-0.2684501	0.4430623	PA	BO
geodist	-0.3000000	0.2621	-0.6594637	0.0524753	PA	BO
Intercept	0.0000000	0.6381	-0.1364747	0.1364747	PA	BO
nlcd_urban_pct	0.0400000	0.7919	-0.1081361	0.1861512	PA	BO
soiltemp_Apr	0.3700000	0.242	0.1301454	0.6194214	PA	BO
soiltemp_Jul	-0.4200000	0.1933	-0.6689095	-0.1809194	PA	BO
R-Squared:	0.1112364	NA	NA	NA	PA	BO
F-Statistic:	3.2541242	NA	NA	NA	PA	BO
F p-value:	0.4281000	NA	NA	NA	PA	BO
distance_to_city_center	-0.1700000	0.4196	-0.4420550	0.0970295	PA	LA
geodist	-0.3100000	0.3217	-0.6281821	0.0014512	PA	LA
Intercept	0.0000000	0.1284	-0.2373979	0.2373979	PA	LA
nlcd_urban_pct	-0.0800000	0.5967	-0.3260509	0.1649127	PA	LA
soiltemp_Apr	0.3500000	0.199	0.0456065	0.6604150	PA	LA
soiltemp_Jul	0.3700000	0.0967	0.1195235	0.6164178	PA	LA
R-Squared:	0.2081501	NA	NA	NA	PA	LA
F-Statistic:	2.0503515	NA	NA	NA	PA	LA
F p-value:	0.3566000	NA	NA	NA	PA	LA
distance_to_city_center	-0.4300000	0.1466	-0.7385505	-0.1244493	PA	PX
geodist	-0.0100000	0.9765	-0.2968494	0.2785857	PA	PX

var	estimate	p	95% Lower	95% Upper	spp	city
Intercept	0.0000000	0.5712	-0.2643406	0.2643406	PA	PX
nlcd_urban_pct	-0.2300000	0.3265	-0.5063135	0.0498969	PA	PX
soiltemp_Apr	0.4900000	0.238	-0.0261493	1.0115755	PA	PX
soiltemp_Jul	-0.2400000	0.4735	-0.7503395	0.2701989	PA	PX
R-Squared:	0.2515098	NA	NA	NA	PA	PX
F-Statistic:	2.0161370	NA	NA	NA	PA	PX
F p-value:	0.4669000	NA	NA	NA	PA	PX
distance_to_city_center	0.0400000	0.8237	-0.1496871	0.2320290	TO	BA
geodist	0.0900000	0.6481	-0.0894511	0.2776567	TO	BA
Intercept	0.0000000	**0.0351**	-0.1341340	0.1341340	TO	BA
nlcd_urban_pct	-0.0800000	0.5617	-0.2223524	0.0531241	TO	BA
soiltemp_Apr	-0.0700000	0.7794	-0.3409639	0.1955214	TO	BA
soiltemp_Jul	0.0000000	0.9954	-0.2736625	0.2770052	TO	BA
R-Squared:	0.0283788	NA	NA	NA	TO	BA
F-Statistic:	0.8587052	NA	NA	NA	TO	BA
F p-value:	0.9491000	NA	NA	NA	TO	BA
distance_to_city_center	0.1200000	0.6929	-0.2202054	0.4582850	TO	BO
geodist	-0.1700000	0.4735	-0.5094324	0.1611152	TO	BO
Intercept	0.0000000	0.6834	-0.1388320	0.1388320	TO	BO
nlcd_urban_pct	-0.0700000	0.5398	-0.2185730	0.0705149	TO	BO
soiltemp_Apr	0.1500000	0.5478	-0.0800745	0.3779286	TO	BO
soiltemp_Jul	0.1200000	0.6453	-0.1132164	0.3464497	TO	BO
R-Squared:	0.0802685	NA	NA	NA	TO	BO
F-Statistic:	2.2691187	NA	NA	NA	TO	BO
F p-value:	0.5396000	NA	NA	NA	TO	BO
distance_to_city_center	0.0500000	0.8054	-0.2219690	0.3168349	TO	LA
geodist	-0.2800000	0.1993	-0.5770585	0.0148853	TO	LA
Intercept	0.0000000	0.4536	-0.1351200	0.1351200	TO	LA
nlcd_urban_pct	0.2300000	0.1091	0.0863749	0.3820454	TO	LA
soiltemp_Apr	-0.6200000	0.0695	-0.7730233	-0.4766983	TO	LA
soiltemp_Jul	0.1800000	0.477	-0.0040807	0.3544541	TO	LA
R-Squared:	0.6014791	NA	NA	NA	TO	LA
F-Statistic:	18.1113470	NA	NA	NA	TO	LA
F p-value:	0.0378000	NA	NA	NA	TO	LA
distance_to_city_center	-0.0500000	0.7679	-0.2175917	0.1139949	TO	MN
geodist	0.0900000	0.5078	-0.0648291	0.2546336	TO	MN
Intercept	0.0000000	0.3552	-0.1116367	0.1116367	TO	MN
nlcd_urban_pct	0.1100000	0.3785	-0.0023696	0.2223177	TO	MN
soiltemp_Apr	0.3200000	0.1515	0.1093125	0.5216227	TO	MN
soiltemp_Jul	-0.3800000	0.0829	-0.5878334	-0.1814124	TO	MN
R-Squared:	0.0643590	NA	NA	NA	TO	MN
F-Statistic:	2.8064698	NA	NA	NA	TO	MN
F p-value:	0.4602000	NA	NA	NA	TO	MN
distance_to_city_center	-0.3300000	0.0867	-0.5879116	-0.0709080	TO	PX
geodist	-0.0700000	0.7911	-0.3994336	0.2535392	TO	PX
Intercept	0.0000000	0.9972	-0.2096401	0.2096401	TO	PX
nlcd_urban_pct	0.1000000	0.6242	-0.1739407	0.3822970	TO	PX
soiltemp_Apr	1.0500000	**0.0245**	0.6584506	1.4482258	TO	PX
soiltemp_Jul	-0.2200000	0.5605	-0.6193773	0.1822039	TO	PX
R-Squared:	0.6599428	NA	NA	NA	TO	PX
F-Statistic:	8.5389990	NA	NA	NA	TO	PX
F p-value:	0.0319000	NA	NA	NA	TO	PX

6 Appendix

6.1 SessionInfo()

```r
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.4.1
## 
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK ve
## 
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## 
## time zone: America/New_York
## tzcode source: internal
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
##  [1] adegenet_2.1.10   ade4_1.7-22       algatr_1.0.0      LEA_3.16.0       
##  [5] ggh4x_0.2.8       here_1.0.1        lubridate_1.9.3   forcats_1.0.0    
##  [9] purrr_1.0.2       tibble_3.2.1      tidyverse_2.0.0   polysat_1.7-7    
## [13] cowplot_1.1.3     viridis_0.6.5     viridisLite_0.4.2 raster_3.6-26    
## [17] sp_2.1-4          stringr_1.5.1     readr_2.1.5       polyRAD_2.0.0    
## [21] dplyr_1.1.4       magrittr_2.0.3    tidyr_1.3.1       ggplot2_3.5.1    
## 
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1  farver_2.1.2      fastmap_1.2.0     promises_1.3.0   
##  [5] digest_0.6.35     mime_0.12         timechange_0.3.0  lifecycle_1.0.4  
##  [9] cluster_2.1.6     terra_1.7-78      compiler_4.4.2    rlang_1.1.4      
## [13] tools_4.4.2       igraph_2.0.3      utf8_1.2.4        yaml_2.3.8       
## [17] knitr_1.47        labeling_0.4.3    bit_4.0.5         plyr_1.8.9       
## [21] xml2_1.3.6        withr_3.0.0       grid_4.4.2        fansi_1.0.6      
## [25] xtable_1.8-4      colorspace_2.1-0  scales_1.3.0      MASS_7.3-61      
## [29] tinytex_0.51      cli_3.6.3         rmarkdown_2.27    vegan_2.6-6.1    
## [33] crayon_1.5.2      ragg_1.3.2        generics_0.1.3    rstudioapi_0.16.0
## [37] reshape2_1.4.4    tzdb_0.4.0        ape_5.8           splines_4.4.2    
## [41] parallel_4.4.2    formatR_1.14      vctrs_0.6.5       Matrix_1.7-1     
## [45] hms_1.1.3         bit64_4.0.5       seqinr_4.2-36     systemfonts_1.1.0
## [49] glue_1.8.0        codetools_0.2-20  stringi_1.8.4     gtable_0.3.5     
## [53] later_1.3.2       munsell_0.5.1     pillar_1.9.0      htmltools_0.5.8.1
## [57] R6_2.5.1          textshaping_0.4.0 rprojroot_2.0.4   shiny_1.8.1.1    
## [61] vroom_1.6.5       evaluate_0.24.0   kableExtra_1.4.0  lattice_0.22-6   
## [65] highr_0.11        httpuv_1.6.15     Rcpp_1.0.12       fastmatch_1.1-4  
## [69] svglite_2.1.3     gridExtra_2.3     nlme_3.1-166      permute_0.9-7    
## [73] mgcv_1.9-1        xfun_0.44         pkgconfig_2.0.3
```

6.2 File Organization

All data files for the Macrosystems project are permanently stored under Meghan Avolio's group resources in the Johns Hopkins University Rockfish computing cluster. Files are stored under the 'data' directory under the following subdirectories:

- `01-raw_data`: This folder contains the raw, unprocessed data files that were obtained directly from the sequencing server. There are eight `fastq.gz` files per sublibrary that correspond to the four sequencing lanes for each read direction.
- `02-concatenated_data`: This folder contains the concatenated, unprocessed files for each sublibrary (i.e., the files containing the sequences for each lane were combined to create one file per read direction).
- `03-pcr_filtered_data`: Here, you will find the resulting data files from the `clone_filter` program, where pcr replicates/clones have been removed from the raw sequences. There are two `fq.gz` files per sublibrary.
- `04-process_radtags`: This folder contains various subdirectories that correspond to the `process_radtags` program that demultiplexes and cleans the data. The `demux_txt_files` folder contains the .txt files used to identify barcodes and separate out the individual samples from each sublibrary. The resulting data files from the `process-radtags` program are separated by individual and can be found in the relevant species folder(i.e., CD, DS, EC, LS, PA, TE, TO). Each individual sample has four data files; `sampleID.1.fq.gz` and `sampleID.2.fq.gz` correspond to the forward and reverse reads for each sample and `sampleID.rem,1/2.fq.gz` contain the remainder reads that were cleaned and removed from the data sequence.
- `05-ustacks-denovo_data`: This folder contains species subdirectories that store the resulting data files from the `ustacks` program for each individual. There are three files per individual; `sampleID.allelles.tsv.gz`, `sampleID.snps.tsv.gz`, and, `sampleID.tags.tsv.gz`. These files should be permanently stored here and copied to a new directory for any new catalogs and/or when a group of samples are being aligned to a new catalog.
- `catalogs_by_city`: For any given species within city, there is likely to be a slightly different set of SNPs compared to the whole metapopulation of five cities. We examined 24 sets of species-city combinations. These catalogs are permanently stored here.
- `catalogs_by_species`: Metapopulation catalogs are stored within this folder for each species. The metapopulation catalog was created using samples from all populations to create a national catalog.

  Some notes about catalog directories:

  - Catalogs contain three files; `catalog.alleles.tsv.gz`, `catalog.snps.tsv.gz`, and `catalog.tafs.tsv.gz`. If you would like to use the catalog on a new project, you will need to copy all three files to a new project folder.
  - You can determine which individuals were used to create the catalog by looking at the `cstacks_popmap.txt` found within each folder. Specifically for the metapopulation catalogs, this information is also found in the cstacks-metapop-catalog_samples-included.csv
  - You can determine which individuals were subsequently aligned to the catalog and used in the subsequent stacks analysis by looking at the `popmap*.txt` found within each folder.
  - Each folder also contains the relevant ustacks and stacks pipeline scripts and output files (i.e., from `cstacks`, `gstacks`, `stacks`, `tsv2bam`, and `populations`),

6.3 Aspera Transfer File Names

See data/aspera_transfer_file_names.csv. Preview:

```r
readLines("data/aspera_transfer_file_names.csv", 10)
```

```
##  [1] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L001_R1_001.fastq.gz"
##  [2] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L001_R2_001.fastq.gz"
##  [3] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L002_R1_001.fastq.gz"
##  [4] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L002_R2_001.fastq.gz"
##  [5] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L003_R1_001.fastq.gz"
##  [6] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L003_R2_001.fastq.gz"
##  [7] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L004_R1_001.fastq.gz"
##  [8] "/Hoffman_macrosystems/AMH_macro_1_1_12px_S1_L004_R2_001.fastq.gz"
##  [9] "/Hoffman_macrosystems/AMH_macro_1_10_8px_S10_L001_R1_001.fastq.gz"
## [10] "/Hoffman_macrosystems/AMH_macro_1_10_8px_S10_L001_R2_001.fastq.gz"
```

6.4 clone_filter File Names

See data/clone_filter_file_names.csv. Preview:

```
readLines("data/clone_filter_file_names.csv", 10)
```

```
##  [1] "AMH_macro_1_1_12px_S1"  "AMH_macro_1_10_8px_S10" "AMH_macro_1_11_8px_S11"
##  [4] "AMH_macro_1_12_8px_S12" "AMH_macro_1_13_8px_S13" "AMH_macro_1_14_8px_S14"
##  [7] "AMH_macro_1_2_12px_S2"  "AMH_macro_1_3_12px_S3"  "AMH_macro_1_4_12px_S4"
## [10] "AMH_macro_1_5_8px_S5"
```