Genetic diversity and population structure in cities is not consistent among cosmopolitan plant species

Supplementary Information

Hoffman et al.

Contents

About

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

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.

0.1 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
  ```

0.2 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.

0.3 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

1 File Transfer

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
```

2 File Concatenation and Stacks Installation

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

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_concatenate_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.

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 PCR Clone Removal

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

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.

3.2 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()
```

4 Sample Demultiplexing and Filtering

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

4.1 Demultiplex and Filter

Step 4a. 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

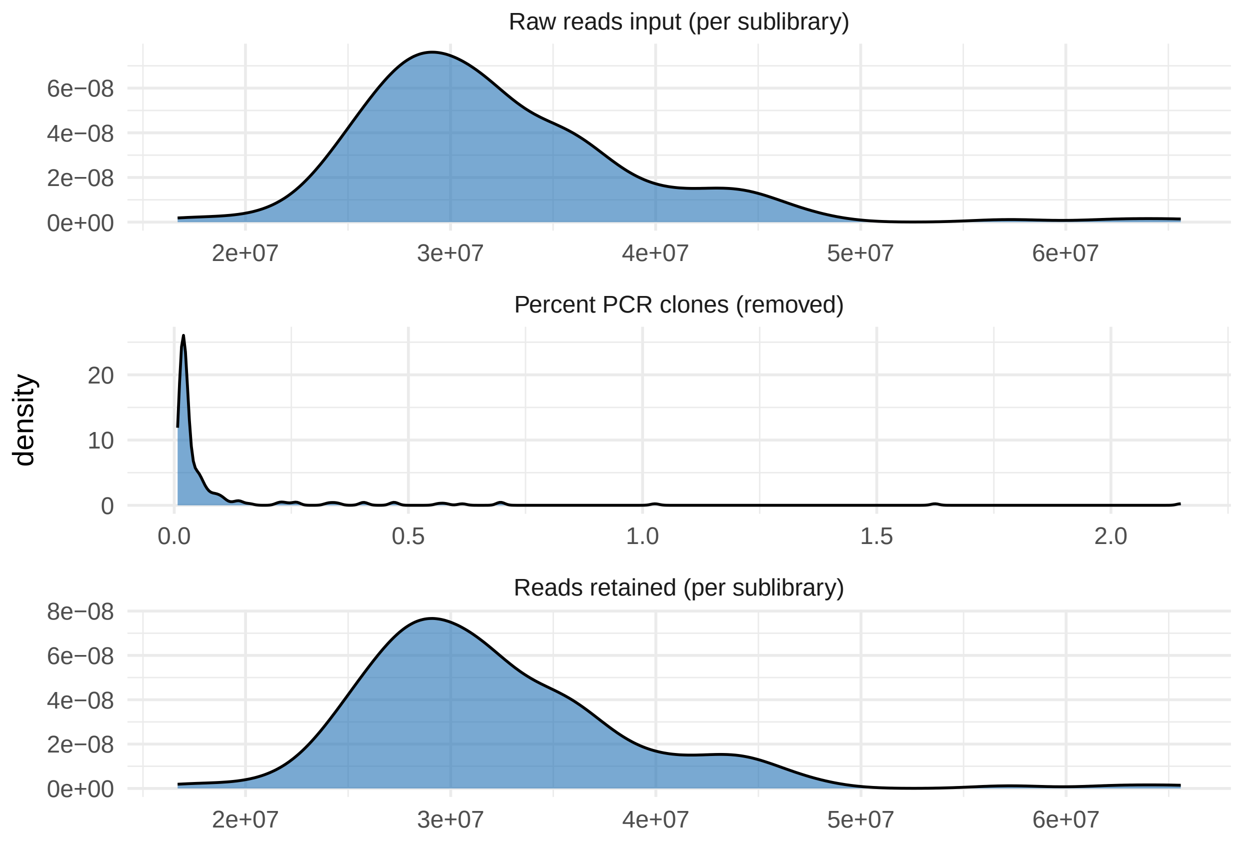

Figure S1: PCR clone removal statistics

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
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.

4.2 Organize files

Step 4b. 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.

4.3 Assess the raw, processed, and cleaned data

Step 4c. In the script for Step 4a, 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()
```

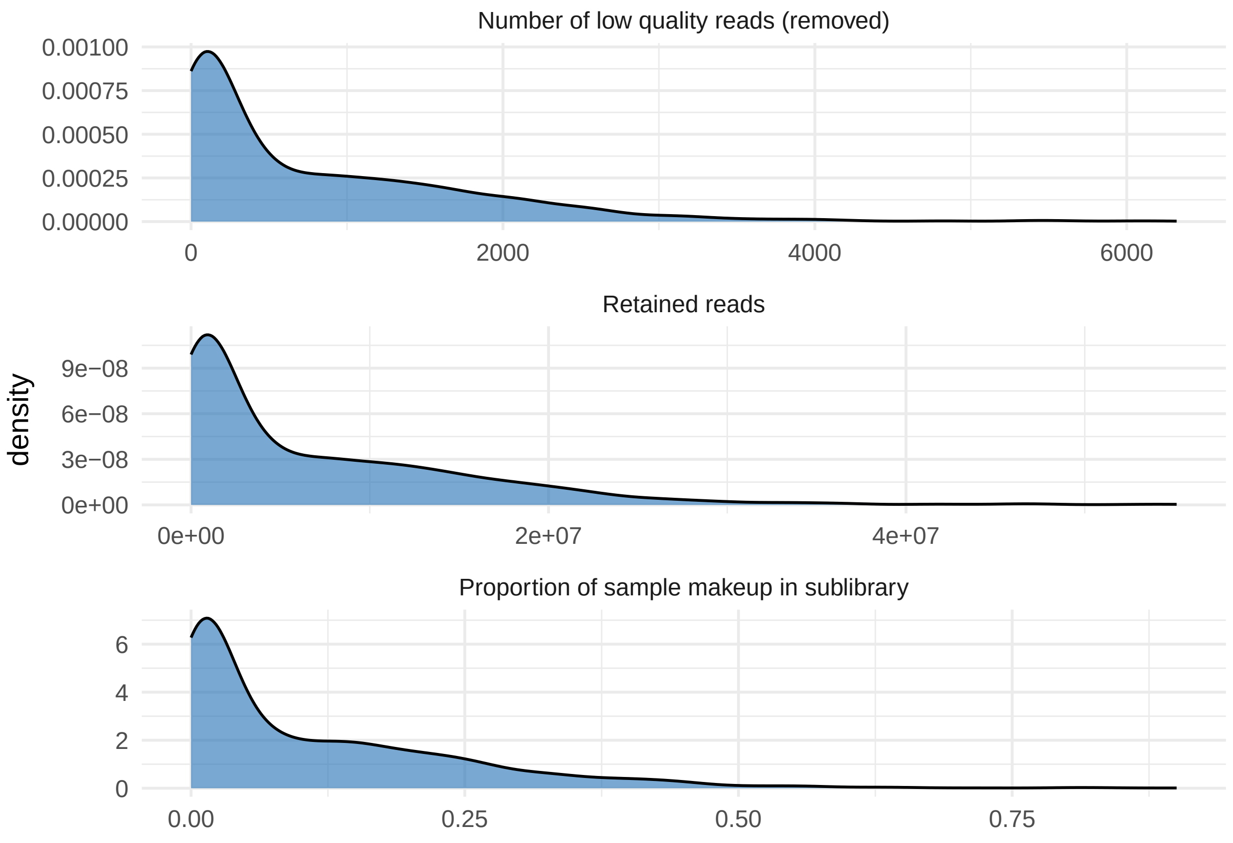

Figure S2: RAD tag processing statistics

4.4 Raw data availability

This is the point at which raw data are available, since it is where they were demultiplexed. Originally, our inventory had 1736 envelopes/samples indexed. Four envelopes/samples had no leaves in them, and

5 samples were the wrong species (*Taraxacum erythrospermum*), and were excluded. We also had one sample that failed to yield any reads (LS.LA.MAR.U.1); it's unclear what happened with this sample (DNA concentration, amplification looked fine - perhaps accidentally given the wrong barcodes). This gives **1726 samples available on SRA**. Note that many of these are excluded due to low coverage, etc., in subsequent steps.

Raw data is available here: https://www.ncbi.nlm.nih.gov/bioproject/PRJNA1359434

4.5 Identify low-coverage and low-quality samples

Step 4d. 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()
```

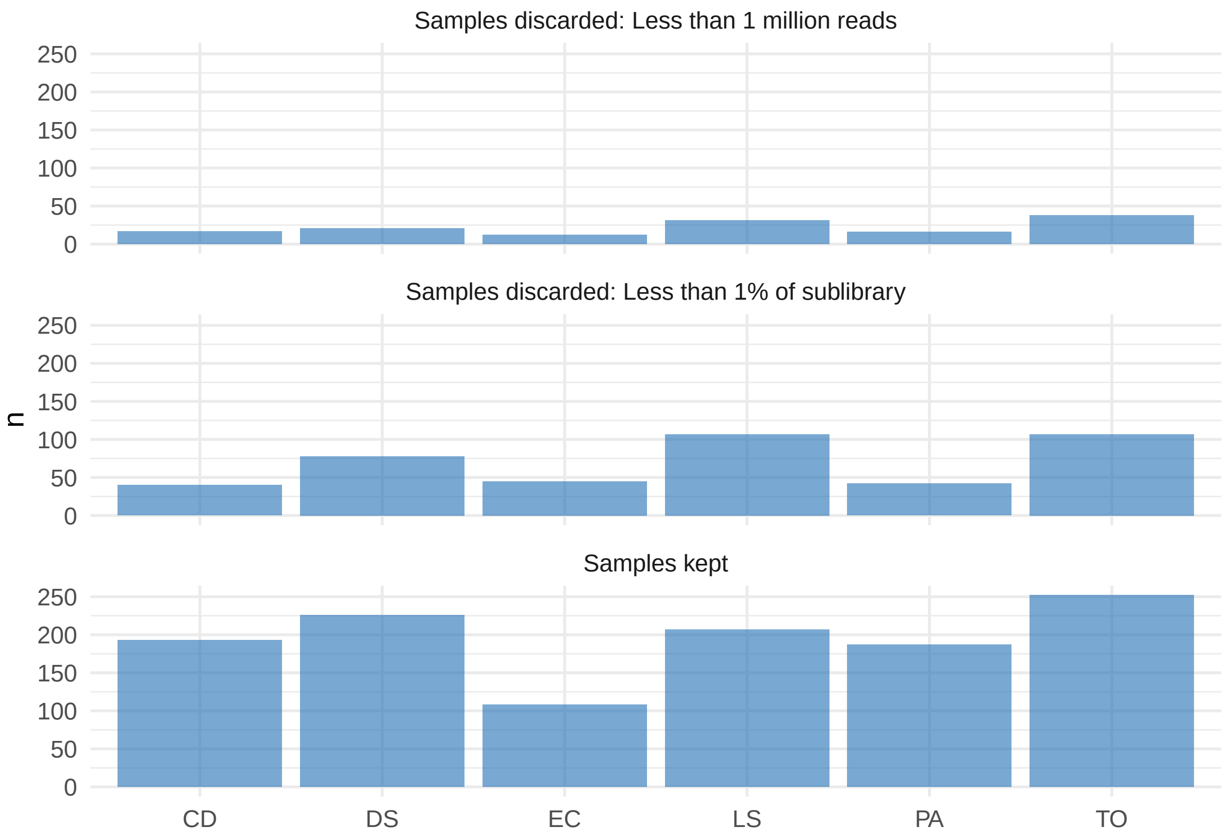

Figure S3: RAD tag manual filtering summary

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.

5 Stacks: 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.

5.1 Run `denovo_map`

Step 5a. 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`:

```
cat 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

5.2 Run `ustacks`

Step 5b. `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 sample 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. Numbers reflect the mean per sample.

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

5.3 Correct File Names

Step 5c. 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. For example, 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.

5.4 Choose catalog samples/files

Step 5d. 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.

6 Stacks: 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

Sample	Species	City
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
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

Sample	Species	City
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
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

Sample	Species	City
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
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

Sample	Species	City
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
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

Sample	Species	City
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
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

7 Stacks: 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.

8 Genotype probabilities with `polyRAD`

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

8.1 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 HPC 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)

8.2 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)

8.3 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)

8.4 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

Sample
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

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

8.5 Data availability

PolyRAD output data is available via Figshare: 10.6084/m9.figshare.30640199, in polyRAD `.rds`, genind `.rds`, and Structure formats.

9 Structure Analysis

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

Structure documentation can be found here.

9.1 Running Structure on `polyRAD` output

`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 this article). Once the optimal K was selected per species, we re-ran Structure using a greater number of iterations (100000) for final output and plotting.

10 Conceptual Figure

We made a conceptual figure to help readers understand PCA patterns we might expect.

```r
source("R/10-Fig1-conceptual_fig.R")
make_fig1()  # Plot Fig 1
```

11 NLCD Data and Site Plots

NLCD is used in Fig. 2 and IBE analysis, described below.

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.

11.1 Preparing NLCD Data

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

```r
source("R/10-trim_NLCD_spatial_data.R")
create_spatial_rds_files()
create_spatial_rds_files(spp = "CD")
```

11.2 Climate normals

We obtained climate normals data for plotting from https://www.ncei.noaa.gov/access/us-climate-normals. We used the latest 30-year period (1991-2020): Most recent standard climatological period (2021 release); which is recommended for most purposes.

11.3 Maps of sampling locations

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

```r
source("R/10-Fig2-plot_map_of_samples.R")
make_all_urban_site_plots_with_clim_normals()  # Plot Fig 2
```

12 Principal components analysis & plots

The following creates PCA plots from polyRAD data.

```
source("R/11-plot_pca.R")
```

```
# Plot Fig 4
make_pca_city_and_pctimp_all()
```

In addition to coloring points by city in the main manuscript, we also colored points by % impervious surface, derived from the NLCD data.

Note that in the pdf version of this document, the figure might appear on the next pages.

```
make_pca_pctimp_only_all()
```

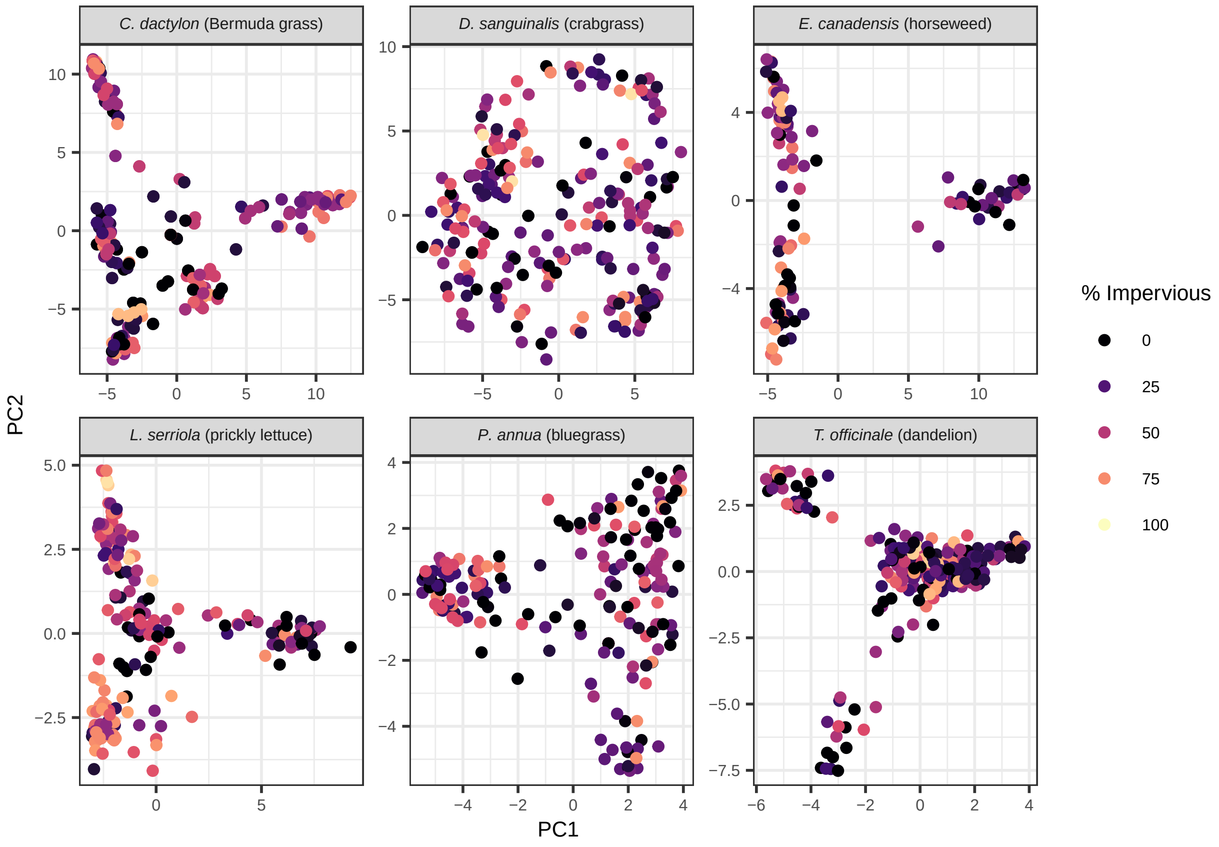

Figure S4: PCA colored by % Impervious surface.

13 Genetic structure using Structure and sNMF

13.1 Structure optimal K

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.

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")
```

The code below generates plots of various K (e.g., K={1-5}) vs likelihood, but did not end up being used in the manuscript.

```
source("R/12-structure_k.R")
```

13.2 Plotting Structure output

The code below generates plots for Structure results.

```
source("R/12-plot_structure.R")
make_structure_multi_plot()  # Plot Fig 5
```

13.3 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/12-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.

```
do_all_sNMF()
```

14 Genetic Differentiation

14.1 Sample size, sites per city per species

We used the following scripts for sample size and sites per city per species.

```
source("R/13-n.R")  # sample size
source("R/13-get_unique_site_post_genotyping.R")  # sites per city per species
```

14.2 Among city - Jost's D, G_{ST}, F_{ST}

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

```
source("R/13-calc_popdiff_stats.R")
```

```
do_all_continental_stats()
```

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

```
##   X statistic      value
## 1 1     JostD 0.30579677
## 2 2       Gst 0.02735719
```

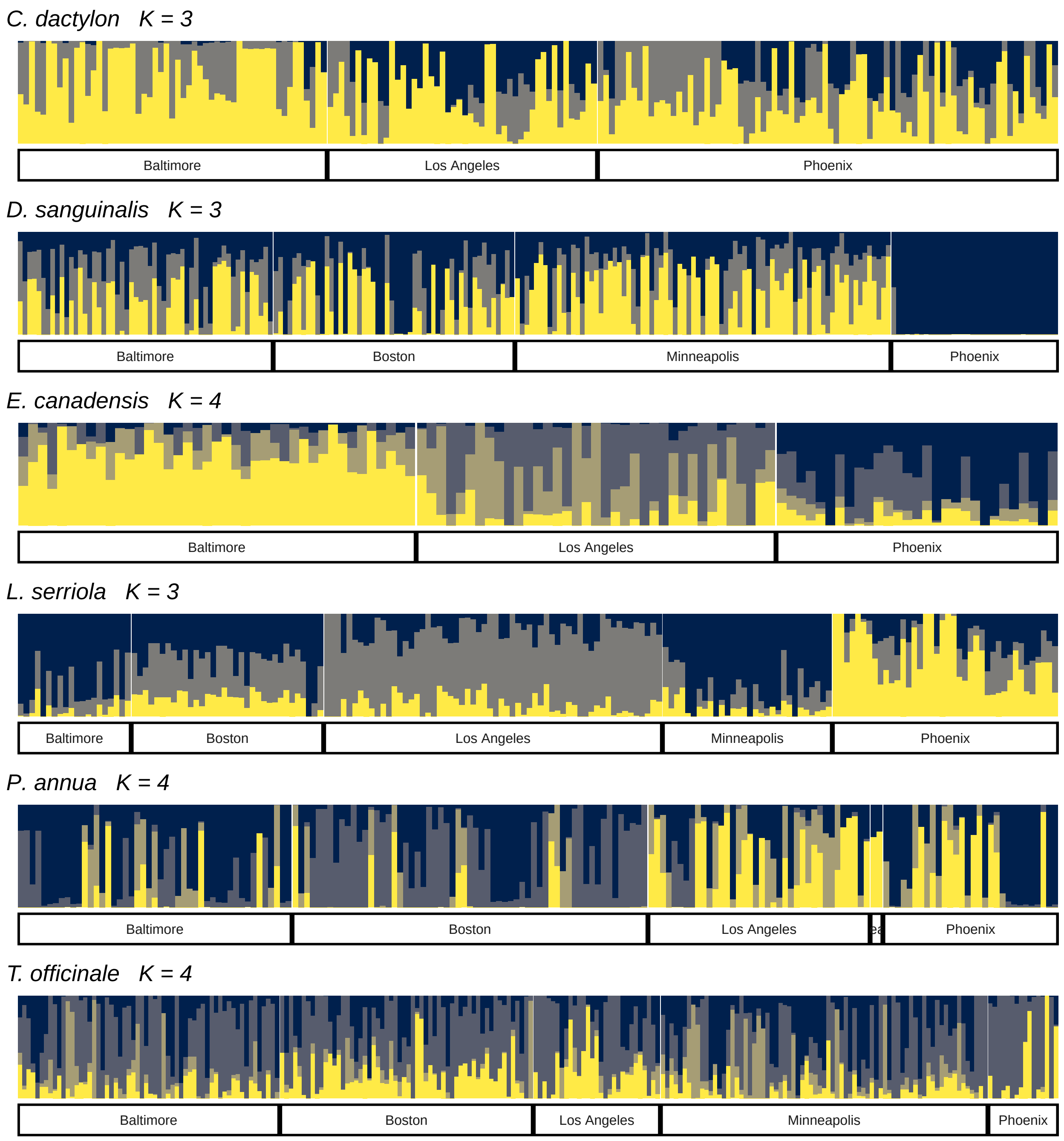

Figure S5: Ancestry coefficients obtained using `snmf()`. As with the Structure analysis, *E. canadensis* (horseweed) and *L. serriola* (prickly lettuce) appear to have the most population structure. *D. sanguinalis*, *E. canadensis*, and *L. serriola* from Phoenix appear unique. In general, sNMF produced larger K for most species, which will create more sensitivity to admixture.

```
## 3 3       Fst 0.02812163
```

14.3 Among city - pairwise ρ

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 Analysis > Pairwise Differentiation and selecting the "rho" statistic from the dropdown.

We used the following script to clean up the results.

```
source("R/13-rho.R")
compile_rho_table()
```

Table S5: Rho statistics for pairwise comparison between cities. Bold+underlined adjusted p-values are significant at the p<0.05 threshold.

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
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**

Species	City1	City2	rho	p-value	adjusted p-value
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

14.4 Within city - allelic richness

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
- 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
```

14.5 Within city - 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.

```
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
```

14.6 Within city - $\bar{r}_d$ - 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.

```
source("R/13-rbarD.R")
calc_rbarD()
```

```
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
```

14.7 Within city - private alleles

We used a custom script to calculate % private alleles as the percentage of total alleles unique to a particular city.

```
source("R/13-private_alleles.R")
get_all_private_alleles()
```

14.8 AMOVA

We performed hierarchical analysis of molecular variance (AMOVA; using GenoDive 3.0.6) 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 excluding these, but found similar results.

Table S6: AMOVA Statistics.

Species	Source of Variation	Nested in	SSD	d.f.	MS	Var-comp	%Var	F-value	P-value
CD	Within Population	–	34441.923	136	253.249	253.249	0.235	0.765	–
CD	Among Population	City	137243.842	46	2983.562	732.212	0.678	0.743	0.001
CD	Among City	–	18722.905	2	9361.452	94.155	0.087	0.087	0.001
DS	Within Population	–	39458.473	155	254.571	254.571	0.224	0.776	–
DS	Among Population	City	149988.556	65	2307.516	638.534	0.561	0.715	0.001
DS	Among City	–	47958.337	3	15986.112	244.451	0.215	0.215	0.001
EC	Within Population	–	15832.596	73	216.885	216.885	0.302	0.698	–

Species	Source of Variation	Nested in	SSD	d.f.	MS	Var-comp	%Var	F-value	P-value
EC	Among Population	City	44957.413	31	1450.239	391.507	0.546	0.644	0.001
EC	Among City	–	10368.036	2	5184.018	108.704	0.152	0.152	0.001
LS	Within Population	–	10889.549	120	90.746	90.746	0.308	0.692	–
LS	Among Population	City	32081.032	59	543.746	159.914	0.544	0.638	0.001
LS	Among City	–	8721.721	4	2180.430	43.569	0.148	0.148	0.001
PA - no MN	Within Population	–	9129.284	128	71.323	71.323	0.230	0.770	–
PA - no MN	Among Population	City	34474.530	44	783.512	194.429	0.628	0.732	0.001
PA - no MN	Among City	–	7968.939	3	2656.313	43.730	0.141	0.141	0.002
PA - w/ MN	Within Population	–	10167.015	129	78.814	78.814	0.278	0.722	–
PA - w/ MN	Among Population	City	30922.681	44	702.788	168.366	0.593	0.681	0.001
PA - w/ MN	Among City	–	6999.878	4	1749.970	36.750	0.129	0.129	0.002
TO	Within Population	–	15727.071	162	97.081	97.081	0.314	0.686	–
TO	Among Population	City	51744.430	71	728.795	202.502	0.655	0.676	0.001
TO	Among City	–	4740.183	4	1185.046	9.469	0.031	0.031	0.001

The following code plots the figure in the main manuscript.

```r
source("R/14-AMOVA.R")
make_amova_plot()  # Plot Fig 6
```

15 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).

15.1 Genetic distance

We calculated the genetic dissimilarity matrix (among sites) 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. We used the "*_estimatedgeno_sitesaspops.structure" files so that sites (not cities) were treated here as populations.

15.2 Geographic distance

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

15.3 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.). This is 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(-89.4, 43.07)
micro <- micro_global(loc = loc)
```

Environmental 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`

15.4 Overall MMRR models

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/15-IBD_IBE_MMRR.R")
make_mmrr_plot()  # Plot Fig 7
```

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

Table S7: Overall model p-values from MMRR.

spp	R-Squared:	F-Statistic:	F p-value:	p
CD	0.0295554	7.126583	0.1852	0.1852
DS	0.0442241	21.515710	0.0817	0.0817
EC	0.2707911	40.996947	0.0001	**1e-04**
LS	0.0465956	19.109304	0.0513	0.0513
PA	0.0051090	1.200614	0.8889	0.8889
TO	0.0283096	16.449250	0.2327	0.2327

Table S8: Full parameter estimates and statistics by species from running 9999 permutations ('Reps') via MMRR.

var	estimate	p	95% Lower	95% Upper	spp
distance_to_city_center	0.1200000	0.1375	0.0694968	0.1664468	CD
geodist	-0.0200000	0.9155	-0.2050334	0.1583669	CD
Intercept	0.0000000	0.9433	-0.0473892	0.0473892	CD
nlcd_urban_pct	-0.1100000	0.0614	-0.1590538	-0.0639585	CD
soiltemp_Apr	-0.1300000	0.7102	-0.4542546	0.1920574	CD
soiltemp_Jul	0.1500000	0.4652	-0.0440647	0.3406560	CD
R-Squared:	0.0295554	NA	NA	NA	CD
F-Statistic:	7.1265831	NA	NA	NA	CD
F p-value:	0.1852000	NA	NA	NA	CD
distance_to_city_center	-0.0200000	0.7678	-0.0534529	0.0157892	DS
geodist	0.0400000	0.1746	-0.0126394	0.0981550	DS
Intercept	0.0000000	0.2102	-0.0340591	0.0326544	DS
nlcd_urban_pct	-0.0400000	0.4238	-0.0739547	-0.0072012	DS
soiltemp_Apr	0.1600000	0.614	-0.0547196	0.3687584	DS
soiltemp_Jul	-0.3900000	0.2492	-0.6148273	-0.1702235	DS
R-Squared:	0.0442241	NA	NA	NA	DS
F-Statistic:	21.5157101	NA	NA	NA	DS
F p-value:	0.0817000	NA	NA	NA	DS
distance_to_city_center	-0.0100000	0.8312	-0.0774980	0.0490001	EC
geodist	-0.2500000	**0.0051**	-0.3654204	-0.1282000	EC
Intercept	0.0000000	**7e-04**	-0.0578968	0.0617663	EC
nlcd_urban_pct	0.0400000	0.5146	-0.0242274	0.1006413	EC
soiltemp_Apr	0.1000000	0.3515	-0.1124231	0.3032273	EC
soiltemp_Jul	0.5600000	**2e-04**	0.4082284	0.7097679	EC
R-Squared:	0.2707911	NA	NA	NA	EC
F-Statistic:	40.9969475	NA	NA	NA	EC
F p-value:	0.0001000	NA	NA	NA	EC
distance_to_city_center	-0.1400000	0.1218	-0.1757714	-0.0999720	LS
geodist	-0.2000000	**0.0014**	-0.2598413	-0.1496999	LS
Intercept	0.0000000	0.0561	-0.0389433	0.0337495	LS
nlcd_urban_pct	-0.0100000	0.8394	-0.0467749	0.0276001	LS
soiltemp_Apr	0.0800000	0.4509	-0.0127788	0.1821182	LS
soiltemp_Jul	0.0500000	0.6937	-0.0276419	0.1341635	LS
R-Squared:	0.0465956	NA	NA	NA	LS
F-Statistic:	19.1093037	NA	NA	NA	LS
F p-value:	0.0513000	NA	NA	NA	LS
distance_to_city_center	0.0500000	0.5062	-0.0008095	0.0955394	PA
geodist	0.0200000	0.8645	-0.1193973	0.1664705	PA
Intercept	0.0000000	0.9891	-0.0480046	0.0480012	PA
nlcd_urban_pct	0.0400000	0.3853	-0.0061648	0.0899176	PA
soiltemp_Apr	-0.0700000	0.7756	-0.3417075	0.2071037	PA
soiltemp_Jul	0.0800000	0.7054	-0.1025172	0.2564145	PA
R-Squared:	0.0051090	NA	NA	NA	PA
F-Statistic:	1.2006139	NA	NA	NA	PA
F p-value:	0.8889000	NA	NA	NA	PA
distance_to_city_center	0.0100000	0.856	-0.0165192	0.0450751	TO
geodist	-0.1100000	**0.044**	-0.1579717	-0.0673055	TO
Intercept	0.0000000	0.6465	-0.0308818	0.0301618	TO
nlcd_urban_pct	0.0100000	0.7993	-0.0163475	0.0452622	TO

var	estimate	p	95% Lower	95% Upper	spp
soiltemp_Apr	0.2700000	0.1202	0.1743641	0.3667842	TO
soiltemp_Jul	-0.3200000	0.1184	-0.4043787	-0.2342648	TO
R-Squared:	0.0283096	NA	NA	NA	TO
F-Statistic:	16.4492496	NA	NA	NA	TO
F p-value:	0.2327000	NA	NA	NA	TO

15.5 MMRR models within city

We also repeated this within city.

Table S9: 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	0.2161593	5.4602354	0.1124	0.1686
CD	LA	0.1221912	1.6704036	0.5615	0.5615
CD	PX	0.2671775	16.4064136	0.0141	**0.0423**
DS	BA	0.0730573	2.0491996	0.6366	0.9613
DS	BO	0.0428518	1.1550721	0.7293	0.9613
DS	MN	0.0239167	1.2104360	0.8725	0.9613
DS	PX	0.0301349	0.3355696	0.9613	0.9613
EC	BA	0.0801989	1.2555586	0.5154	0.5154
EC	LA	0.1542581	1.4226717	0.4961	0.5154
EC	PX	0.1802413	2.1107626	0.3254	0.5154
LS	BA	0.6200687	6.8546312	0.1892	0.473
LS	BO	0.0447270	0.7397752	0.8838	0.8838
LS	LA	0.1902178	6.9060584	0.1242	0.473
LS	MN	0.1245043	1.5358687	0.4772	0.7953
LS	PX	0.0551478	0.7003986	0.8244	0.8838
PA	BA	0.1429473	2.0014720	0.2072	0.4144
PA	BO	0.0405967	1.1001781	0.7435	0.8569
PA	LA	0.0752459	0.6346749	0.8569	0.8569
PA	PX	0.5738665	8.0800948	0.0427	0.1708
TO	BA	0.0340749	1.0371416	0.8999	0.8999
TO	BO	0.0833431	2.3639386	0.4830	0.7145
TO	LA	0.8796290	83.3072026	0.0003	**0.0015**
TO	MN	0.0887157	3.9719783	0.3268	0.7145
TO	PX	0.2290994	1.1292997	0.5716	0.7145

Table S10: 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	-0.1000000	0.4185	-0.2684666	0.0632947	CD	BA
geodist	0.1800000	0.2432	0.0183554	0.3382114	CD	BA
Intercept	0.0000000	**0.0051**	-0.1470377	0.1470377	CD	BA
nlcd_urban_pct	-0.4200000	**0.001**	-0.5689671	-0.2638461	CD	BA
soiltemp_Apr	0.0200000	0.9443	-0.2536769	0.2879531	CD	BA

var	estimate	p	95% Lower	95% Upper	spp	city
soiltemp_Jul	0.1300000	0.5869	-0.1372739	0.4066275	CD	BA
R-Squared:	0.2161593	NA	NA	NA	CD	BA
F-Statistic:	5.4602354	NA	NA	NA	CD	BA
F p-value:	0.1124000	NA	NA	NA	CD	BA
distance_to_city_center	0.2900000	0.1362	0.0584736	0.5275481	CD	LA
geodist	-0.3300000	0.2694	-0.6122966	-0.0485802	CD	LA
Intercept	0.0000000	0.3495	-0.2005369	0.2005369	CD	LA
nlcd_urban_pct	-0.0300000	0.8776	-0.2309849	0.1796022	CD	LA
soiltemp_Apr	-0.0700000	0.7776	-0.3389143	0.2056573	CD	LA
soiltemp_Jul	0.1000000	0.6473	-0.1130946	0.3099422	CD	LA
R-Squared:	0.1221912	NA	NA	NA	CD	LA
F-Statistic:	1.6704036	NA	NA	NA	CD	LA
F p-value:	0.5615000	NA	NA	NA	CD	LA
distance_to_city_center	0.5400000	**0.0112**	0.4048935	0.6809183	CD	PX
geodist	0.2400000	0.1288	0.1176737	0.3580622	CD	PX
Intercept	0.0000000	0.8859	-0.0940557	0.0940557	CD	PX
nlcd_urban_pct	-0.0200000	0.8422	-0.1204062	0.0794245	CD	PX
soiltemp_Apr	-0.3400000	0.1006	-0.5503081	-0.1239513	CD	PX
soiltemp_Jul	-0.0700000	0.6137	-0.2703666	0.1311920	CD	PX
R-Squared:	0.2671775	NA	NA	NA	CD	PX
F-Statistic:	16.4064136	NA	NA	NA	CD	PX
F p-value:	0.0141000	NA	NA	NA	CD	PX
distance_to_city_center	-0.0800000	0.6566	-0.2519155	0.0908257	DS	BA
geodist	0.2100000	0.1108	0.0335974	0.3810348	DS	BA
Intercept	0.0000000	0.761	-0.1393752	0.1393752	DS	BA
nlcd_urban_pct	-0.1500000	0.2505	-0.2857403	-0.0045114	DS	BA
soiltemp_Apr	-0.2200000	0.4722	-0.4847959	0.0528128	DS	BA
soiltemp_Jul	0.2900000	0.3347	0.0227011	0.5523010	DS	BA
R-Squared:	0.0730573	NA	NA	NA	DS	BA
F-Statistic:	2.0491996	NA	NA	NA	DS	BA
F p-value:	0.6366000	NA	NA	NA	DS	BA
distance_to_city_center	0.4700000	0.1593	0.1137323	0.8172063	DS	BO
geodist	-0.4500000	0.0645	-0.7938183	-0.0989208	DS	BO
Intercept	0.0000000	0.8498	-0.1425615	0.1418102	DS	BO
nlcd_urban_pct	0.0500000	0.7466	-0.1048400	0.1967690	DS	BO
soiltemp_Apr	-0.1200000	0.6561	-0.3734660	0.1401180	DS	BO
soiltemp_Jul	0.1400000	0.5493	-0.1201335	0.3927905	DS	BO
R-Squared:	0.0428518	NA	NA	NA	DS	BO
F-Statistic:	1.1550721	NA	NA	NA	DS	BO
F p-value:	0.7293000	NA	NA	NA	DS	BO
distance_to_city_center	-0.0400000	0.7746	-0.1840544	0.1117149	DS	MN
geodist	-0.0400000	0.7876	-0.1818746	0.1105641	DS	MN
Intercept	0.0000000	0.4679	-0.1035844	0.1035844	DS	MN
nlcd_urban_pct	0.0400000	0.699	-0.0727871	0.1502068	DS	MN
soiltemp_Apr	-0.1800000	0.4033	-0.3555524	-0.0143666	DS	MN
soiltemp_Jul	0.0600000	0.7516	-0.1097516	0.2385213	DS	MN
R-Squared:	0.0239167	NA	NA	NA	DS	MN
F-Statistic:	1.2104360	NA	NA	NA	DS	MN
F p-value:	0.8725000	NA	NA	NA	DS	MN
distance_to_city_center	-0.0300000	0.9108	-0.3758396	0.3139799	DS	PX
geodist	-0.0800000	0.7542	-0.3420557	0.1911879	DS	PX
Intercept	0.0100000	0.6325	-0.2137801	0.2332494	DS	PX

var	estimate	p	95% Lower	95% Upper	spp	city
nlcd_urban_pct	0.0800000	0.6269	-0.1509445	0.3132693	DS	PX
soiltemp_Apr	-0.0800000	0.8003	-0.4400966	0.2847726	DS	PX
soiltemp_Jul	-0.0600000	0.7774	-0.3749012	0.2644566	DS	PX
R-Squared:	0.0301349	NA	NA	NA	DS	PX
F-Statistic:	0.3355696	NA	NA	NA	DS	PX
F p-value:	0.9613000	NA	NA	NA	DS	PX
distance_to_city_center	0.1000000	0.5104	-0.1352494	0.3274849	EC	BA
geodist	0.1700000	0.2716	-0.0622310	0.3983733	EC	BA
Intercept	0.0000000	0.6783	-0.1871244	0.1871244	EC	BA
nlcd_urban_pct	0.1200000	0.3852	-0.0792998	0.3107074	EC	BA
soiltemp_Apr	0.0500000	0.8657	-0.2985420	0.3896308	EC	BA
soiltemp_Jul	0.0600000	0.8204	-0.2871230	0.3984088	EC	BA
R-Squared:	0.0801989	NA	NA	NA	EC	BA
F-Statistic:	1.2555586	NA	NA	NA	EC	BA
F p-value:	0.5154000	NA	NA	NA	EC	BA
distance_to_city_center	-0.9800000	0.0737	-1.6450191	-0.3099034	EC	LA
geodist	0.2400000	0.2494	-0.0685768	0.5511398	EC	LA
Intercept	0.0000000	0.3457	-0.2453434	0.2453434	EC	LA
nlcd_urban_pct	0.5500000	0.1414	0.0320693	1.0648316	EC	LA
soiltemp_Apr	-0.0200000	0.9446	-0.2807172	0.2483734	EC	LA
soiltemp_Jul	0.4200000	0.232	0.0302705	0.8043884	EC	LA
R-Squared:	0.1542581	NA	NA	NA	EC	LA
F-Statistic:	1.4226717	NA	NA	NA	EC	LA
F p-value:	0.4961000	NA	NA	NA	EC	LA
distance_to_city_center	0.0100000	0.9703	-0.2834277	0.3034585	EC	PX
geodist	-0.3900000	**<u>0.0395</u>**	-0.6389291	-0.1363801	EC	PX
Intercept	0.0100000	0.3856	-0.2107459	0.2237152	EC	PX
nlcd_urban_pct	0.0000000	0.9772	-0.2250492	0.2151403	EC	PX
soiltemp_Apr	-0.0100000	0.9779	-0.4668132	0.4495769	EC	PX
soiltemp_Jul	-0.0900000	0.7174	-0.5232281	0.3458628	EC	PX
R-Squared:	0.1802413	NA	NA	NA	EC	PX
F-Statistic:	2.1107626	NA	NA	NA	EC	PX
F p-value:	0.3254000	NA	NA	NA	EC	PX
distance_to_city_center	1.1900000	0.106	0.7662079	1.6094443	LS	BA
geodist	-1.0400000	0.1159	-1.4829287	-0.6016138	LS	BA
Intercept	-0.0300000	0.1484	-0.2622639	0.1947604	LS	BA
nlcd_urban_pct	0.2700000	0.2101	0.0266277	0.5221911	LS	BA
soiltemp_Apr	0.6400000	0.1682	0.2034475	1.0693071	LS	BA
soiltemp_Jul	-0.5100000	0.1641	-0.9457492	-0.0759057	LS	BA
R-Squared:	0.6200687	NA	NA	NA	LS	BA
F-Statistic:	6.8546312	NA	NA	NA	LS	BA
F p-value:	0.1892000	NA	NA	NA	LS	BA
distance_to_city_center	-0.2600000	0.5926	-0.7986157	0.2851344	LS	BO
geodist	0.3500000	0.3781	-0.1877151	0.8820930	LS	BO
Intercept	0.0000000	0.7776	-0.1855937	0.1791482	LS	BO
nlcd_urban_pct	-0.0500000	0.7507	-0.2336700	0.1374681	LS	BO
soiltemp_Apr	0.3100000	0.3858	-0.0610043	0.6888627	LS	BO
soiltemp_Jul	-0.3500000	0.3456	-0.7122678	0.0129835	LS	BO
R-Squared:	0.0447270	NA	NA	NA	LS	BO
F-Statistic:	0.7397752	NA	NA	NA	LS	BO
F p-value:	0.8838000	NA	NA	NA	LS	BO
distance_to_city_center	0.0000000	0.9992	-0.2315808	0.2320620	LS	LA

var	estimate	p	95% Lower	95% Upper	spp	city
geodist	-0.3500000	0.1233	-0.5764863	-0.1159599	LS	LA
Intercept	0.0000000	0.0865	-0.1224544	0.1224544	LS	LA
nlcd_urban_pct	-0.0900000	0.3924	-0.2277126	0.0497195	LS	LA
soiltemp_Apr	0.1300000	0.2848	-0.0039581	0.2738997	LS	LA
soiltemp_Jul	0.3300000	**0.0401**	0.2024289	0.4634342	LS	LA
R-Squared:	0.1902178	NA	NA	NA	LS	LA
F-Statistic:	6.9060584	NA	NA	NA	LS	LA
F p-value:	0.1242000	NA	NA	NA	LS	LA
distance_to_city_center	-0.2200000	0.2576	-0.4522034	0.0138133	LS	MN
geodist	0.2200000	0.2056	-0.0135483	0.4484999	LS	MN
Intercept	0.0000000	0.7834	-0.2081530	0.2173090	LS	MN
nlcd_urban_pct	0.2000000	0.2129	-0.0369112	0.4346262	LS	MN
soiltemp_Apr	-0.0700000	0.828	-0.4214409	0.2856407	LS	MN
soiltemp_Jul	0.0500000	0.8648	-0.3132217	0.4139720	LS	MN
R-Squared:	0.1245043	NA	NA	NA	LS	MN
F-Statistic:	1.5358687	NA	NA	NA	LS	MN
F p-value:	0.4772000	NA	NA	NA	LS	MN
distance_to_city_center	0.0800000	0.6914	-0.2026496	0.3675307	LS	PX
geodist	-0.1000000	0.7047	-0.4513560	0.2458115	LS	PX
Intercept	0.0000000	0.8133	-0.2080541	0.2080541	LS	PX
nlcd_urban_pct	0.1600000	0.2157	-0.0485440	0.3744609	LS	PX
soiltemp_Apr	-0.3000000	0.664	-1.1674789	0.5580362	LS	PX
soiltemp_Jul	0.2100000	0.7654	-0.5359827	0.9536103	LS	PX
R-Squared:	0.0551478	NA	NA	NA	LS	PX
F-Statistic:	0.7003986	NA	NA	NA	LS	PX
F p-value:	0.8244000	NA	NA	NA	LS	PX
distance_to_city_center	-0.2400000	0.1406	-0.4569014	-0.0211339	PA	BA
geodist	0.0400000	0.7215	-0.1864189	0.2760527	PA	BA
Intercept	0.0000000	0.3402	-0.1981518	0.1981518	PA	BA
nlcd_urban_pct	0.3400000	**0.0292**	0.1245412	0.5455189	PA	BA
soiltemp_Apr	0.1200000	0.4757	-0.1805703	0.4123389	PA	BA
soiltemp_Jul	-0.0900000	0.7229	-0.3920738	0.2124508	PA	BA
R-Squared:	0.1429473	NA	NA	NA	PA	BA
F-Statistic:	2.0014720	NA	NA	NA	PA	BA
F p-value:	0.2072000	NA	NA	NA	PA	BA
distance_to_city_center	0.3000000	0.3375	-0.0653382	0.6739094	PA	BO
geodist	-0.2600000	0.3184	-0.6268734	0.1128175	PA	BO
Intercept	0.0000000	0.476	-0.1417946	0.1417946	PA	BO
nlcd_urban_pct	0.1100000	0.396	-0.0432134	0.2625454	PA	BO
soiltemp_Apr	0.0600000	0.8052	-0.1923722	0.3159761	PA	BO
soiltemp_Jul	0.0800000	0.7166	-0.1751357	0.3318766	PA	BO
R-Squared:	0.0405967	NA	NA	NA	PA	BO
F-Statistic:	1.1001781	NA	NA	NA	PA	BO
F p-value:	0.7435000	NA	NA	NA	PA	BO
distance_to_city_center	0.0500000	0.8015	-0.2385409	0.3440297	PA	LA
geodist	-0.0100000	0.9725	-0.3525149	0.3279088	PA	LA
Intercept	0.0000000	0.2965	-0.2565480	0.2565480	PA	LA
nlcd_urban_pct	-0.2300000	0.1389	-0.4996017	0.0309663	PA	LA
soiltemp_Apr	-0.1600000	0.5729	-0.4959573	0.1684456	PA	LA
soiltemp_Jul	-0.0200000	0.9582	-0.2837329	0.2532441	PA	LA
R-Squared:	0.0752459	NA	NA	NA	PA	LA
F-Statistic:	0.6346749	NA	NA	NA	PA	LA

var	estimate	p	95% Lower	95% Upper	spp	city
F p-value:	0.8569000	NA	NA	NA	PA	LA
distance_to_city_center	-0.3400000	0.1264	-0.5736448	-0.1102836	PA	PX
geodist	0.3000000	**<u>0.0395</u>**	0.0831452	0.5173315	PA	PX
Intercept	0.0000000	0.3987	-0.1994544	0.1994544	PA	PX
nlcd_urban_pct	0.0800000	0.4762	-0.1282009	0.2914797	PA	PX
soiltemp_Apr	0.5700000	0.0869	0.1810453	0.9640457	PA	PX
soiltemp_Jul	0.1800000	0.4031	-0.2095945	0.5604381	PA	PX
R-Squared:	0.5738665	NA	NA	NA	PA	PX
F-Statistic:	8.0800948	NA	NA	NA	PA	PX
F p-value:	0.0427000	NA	NA	NA	PA	PX
distance_to_city_center	-0.0600000	0.7548	-0.2527562	0.1278394	TO	BA
geodist	0.1700000	0.4582	-0.0175679	0.3484622	TO	BA
Intercept	0.0000000	0.9428	-0.1337402	0.1337402	TO	BA
nlcd_urban_pct	0.0200000	0.8871	-0.1201015	0.1545664	TO	BA
soiltemp_Apr	0.0200000	0.925	-0.2456204	0.2892900	TO	BA
soiltemp_Jul	-0.1200000	0.6149	-0.3955572	0.1534940	TO	BA
R-Squared:	0.0340749	NA	NA	NA	TO	BA
F-Statistic:	1.0371416	NA	NA	NA	TO	BA
F p-value:	0.8999000	NA	NA	NA	TO	BA
distance_to_city_center	0.2500000	0.4583	-0.0866987	0.5906567	TO	BO
geodist	-0.3200000	0.1915	-0.6505818	0.0188440	TO	BO
Intercept	0.0000000	0.9424	-0.1385997	0.1385997	TO	BO
nlcd_urban_pct	0.2200000	0.056	0.0797738	0.3683782	TO	BO
soiltemp_Apr	0.0200000	0.9385	-0.2101130	0.2471239	TO	BO
soiltemp_Jul	-0.1400000	0.5419	-0.3690613	0.0898358	TO	BO
R-Squared:	0.0833431	NA	NA	NA	TO	BO
F-Statistic:	2.3639386	NA	NA	NA	TO	BO
F p-value:	0.4830000	NA	NA	NA	TO	BO
distance_to_city_center	-0.0600000	0.5889	-0.2085607	0.0944370	TO	LA
geodist	0.0100000	0.9256	-0.1584939	0.1794819	TO	LA
Intercept	-0.0300000	0.11	-0.1062405	0.0465560	TO	LA
nlcd_urban_pct	-0.0300000	0.7391	-0.1111975	0.0553448	TO	LA
soiltemp_Apr	-0.9400000	**<u>2e-04</u>**	-1.0183148	-0.8521554	TO	LA
soiltemp_Jul	0.0600000	0.5799	-0.0386642	0.1617719	TO	LA
R-Squared:	0.8796290	NA	NA	NA	TO	LA
F-Statistic:	83.3072026	NA	NA	NA	TO	LA
F p-value:	0.0003000	NA	NA	NA	TO	LA
distance_to_city_center	-0.0300000	0.8513	-0.1973255	0.1299167	TO	MN
geodist	0.0200000	0.9075	-0.1406409	0.1746362	TO	MN
Intercept	0.0000000	0.6183	-0.1101741	0.1101741	TO	MN
nlcd_urban_pct	0.0400000	0.7863	-0.0699047	0.1518388	TO	MN
soiltemp_Apr	0.4800000	**<u>0.0338</u>**	0.2780902	0.6849983	TO	MN
soiltemp_Jul	-0.3200000	0.1707	-0.5199557	-0.1188595	TO	MN
R-Squared:	0.0887157	NA	NA	NA	TO	MN
F-Statistic:	3.9719783	NA	NA	NA	TO	MN
F p-value:	0.3268000	NA	NA	NA	TO	MN
distance_to_city_center	0.2700000	0.3084	-0.1389457	0.6732755	TO	PX
geodist	0.0300000	0.9349	-0.4735969	0.5318689	TO	PX
Intercept	-0.0100000	0.8142	-0.3570472	0.3342439	TO	PX
nlcd_urban_pct	0.1600000	0.5206	-0.2774734	0.6011279	TO	PX
soiltemp_Apr	-0.5000000	0.3107	-1.1136111	0.1065639	TO	PX
soiltemp_Jul	0.0900000	0.8399	-0.5216801	0.7115915	TO	PX

var	estimate	p	95% Lower	95% Upper	spp	city
R-Squared:	0.2290994	NA	NA	NA	TO	PX
F-Statistic:	1.1292997	NA	NA	NA	TO	PX
F p-value:	0.5716000	NA	NA	NA	TO	PX

16 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 v
##
## 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] polyRAD_2.0.0   polysat_1.7-7   ggh4x_0.2.8     LEA_3.16.0
##  [5] lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1   purrr_1.0.2
##  [9] tibble_3.2.1    tidyverse_2.0.0 cowplot_1.2.0   readr_2.1.5
## [13] here_1.0.1      dplyr_1.1.4     magrittr_2.0.3  tidyr_1.3.1
## [17] ggplot2_4.0.0
##
## loaded via a namespace (and not attached):
##  [1] fastmatch_1.1-4    gtable_0.3.6       xfun_0.52          tzdb_0.4.0
##  [5] vctrs_0.6.5        tools_4.4.2        generics_0.1.3     parallel_4.4.2
##  [9] fansi_1.0.6        highr_0.11         pkgconfig_2.0.3    RColorBrewer_1.1-3
## [13] S7_0.2.0           lifecycle_1.0.4    compiler_4.4.2     farver_2.1.2
## [17] textshaping_0.4.0  tinytex_0.51       htmltools_0.5.8.1  yaml_2.3.10
## [21] pillar_1.9.0       crayon_1.5.2       viridis_0.6.5      commonmark_1.9.1
## [25] tidyselect_1.2.1   digest_0.6.35      stringi_1.8.4      labeling_0.4.3
## [29] rprojroot_2.0.4    fastmap_1.2.0      grid_4.4.2         cli_3.6.3
## [33] dichromat_2.0-0.1  utf8_1.2.4         withr_3.0.2        scales_1.4.0
## [37] bit64_4.0.5        timechange_0.3.0   rmarkdown_2.27     bit_4.0.5
## [41] ggtext_0.1.2       gridExtra_2.3      ragg_1.3.2         hms_1.1.3
## [45] kableExtra_1.4.0   evaluate_1.0.5     knitr_1.47         viridisLite_0.4.2
## [49] markdown_1.13      rlang_1.1.4        gridtext_0.1.5     Rcpp_1.0.12
## [53] glue_1.8.0         formatR_1.14       xml2_1.3.6         svglite_2.1.3
## [57] rstudioapi_0.16.0  vroom_1.6.5        R6_2.5.1           systemfonts_1.1.0
```

Appendix

16.1 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`),

16.2 Aspera Transfer File Names

See [data/aspera_transfer_file_names.csv] (data/aspera_transfer_file_names.csv). Preview:

```
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"
```

16.3 `clone_filter` File Names

See [data/clone__filter__file__names.csv] (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"
```