Documentation for the Ivory Pipeline v1.0

Mary Kuhner, CEFS

Spring 2025

This document contains the practical steps needed to process African elephant microsatellite data, including both incorporating new reference data, and running new seizures. The pipeline should run fairly straightforwardly on Linux or MacOS from the command line. I don't believe it will run as-is on Windows; specification of file paths is likely to need wholesale revision. It was developed on Ubuntu Linux; I have tried to distinguish Ubuntu-specific information from general information.

Please be careful to use a text editor, not a word processor, for any changes to pipeline programs or files.

Table of Contents

1. Concepts

2. Setting up to run pipeline

2A. Needed resources and the ivory\_paths.tsv file

2B. Code and libraries

3. Processing reference data and creating archives

3A. Extracting data from master database

3B. Species identification and hybrid removal

4. Processing seizure data

4A. Extracting data from master database

4B. Validation, species identification, and hybrid removal

4C. Familial matching

4D. SCAT

4E. VORONOI

4F. Report generation

5. Future directions

6. References

**1. Concepts**

Broadly speaking, this pipeline can run initial setup and data prep; species and hybrid identification; SCAT analysis; VORONOI analysis; and familial matching. Instructions are also provided for setting up a reference database to which seizures can be compared.

*Species and hybrid identification.* This is done by EBhybrids (Mondol et al. 2015) using species-identified reference data; initial reference-data species identification is done by STRUCTURE (Pritchard et al. 2000) Currently we define a "hybrid" as a sample with 50% or higher hybrid probability in EBhybrids. Such individuals are reported on and removed from further analysis. We are unable to perform geolocation or familial matching on hybrids due to lack of appropriate control allele frequencies.

*SCAT analysis.* The SCAT program (Wasser et al. 2004) geolocates elephants. While it is run on the whole seizure at once, it localizes each individual in turn using only the reference data, not other elephants in the seizure. Note that the uncertainty of the estimates varies widely and can be enormous. The pipeline generates heatmaps which can help in visualizing uncertainty.

*VORONOI analysis.* The VORONOI program (Wasser et al. 2007) clusters SCAT output to estimate the location of the whole seizure, assuming that seizure elephants are from a restricted range relative to all elephants of their species. When the seizure is in fact a single cluster, this greatly improves accuracy. However, elephants in the seizure that do not belong in the main cluster are often misplaced there. VORONOI analysis does not appear to add value for samples of fewer than 10 elephants of a given species and is not run in those cases. It should also not be run if the data are not a single seizure, for example the contents of a long-term stockpile.

In the near future I hope to replace VORONOI with the CLUST program.

*Familial matching.* The familial matching program (Wasser et al. 2022) is called "fammatch" throughout the pipeline and documentation: Charles Wolock, its author, called the GitHub site "elephant\_fam\_match" but the program itself is a collection of R scripts with no clear name. This program identifies exact and close-relative (parent/offspring, full and half sibs) matches between samples from different seizures (Wasser et al. 2022). Because of the enormous number of elephant pairs to be checked for matching, false-positive rates are high, complicating interpretation of the results.

If your data are not actually a seizure, you should NOT run familial matching on them, as this will insert their information into the seizure database and later seizures will be matched against your non-seizure. A no-familial-matching version of the pipeline is included.

*Interactions.* Data prep and species identification are obligatory for all other steps. SCAT and VORONOI are independent of familial matching, though familial matching does require a SCAT executable. VORONOI relies on SCAT and cannot be run unless SCAT has been run first.

*General concepts and nomenclature.*

The code assumes that the Wasser Lab 16 microsatellite panel is being used, and can also be made to run if only a subset was used (e.g. Julie Bonnald's 15-microsatellite panel). Microsats must be in the canonical Wasser Lab order, which is found in file ivory\_pipeline/aux/header\_for\_structure and is:

FH67 FH71 FH19 FH129 FH60 FH127 FH126 FH153 FH94 FH48 FH40 FH39 FH103 FH102 S03 S04

Most analyses are done one species at a time, keyed off species name: *L. africana* is called "savannah" (though "savanna" might be more technically correct) and *L. cyclotis* is "forest."

A seizure is identified by a "PREFIX" which I recommend should be the formal seizure name, but with spaces removed and commas replaced by underscores. Seizure HKG, 01-13, 1.3t thus becomes HKG\_01-13\_1.3t. This prefix is used in file and directory names and cannot contain spaces, nor punctuation other than hyphen and underscore.

I strongly recommend not running the pipeline until the permanent name of the seizure has been established; if it is run with an incorrect name the output and directory structure will be full of references to that name and this is painful to fix. The script renameseizuredir.py in /ivory\_pipeline/src renames files and directories, but does not catch information inside of files. The best solution to having run a seizure with the wrong name is probably to delete it and start from scratch with the right name.

An elephant is identified by SID. If it's necessary to go from SID to seizure, the file "seizure\_metadata" in the familial matching archive can be used. The familial matching code internally truncates SIDs to 11 characters, so they must be unique in the first 11 characters (all current SIDs obey this rule).

A reference data version is identified by a "REFPREFIX" which I recommend should be REFELE\_version.subversion, for example REFELE\_5.8, which corresponds to master elephant database release 5.8.

The pipeline strongly assumes that all seizure-specific directories are subdirectories under a single directory, called "parent directory of all seizures" throughout this documentation. Except for SCAT runs, all of the pipeline programs expect to be run from this parent directory.

**2. Setting up to run pipeline**

*Seizure and reference data*

Seizure and reference data reside in the master elephant database, an .xlsx file which is maintained by Ada. This is not kept on public sites for data security reasons.

If you do not have pre-processed reference files, you will need a copy of this database from which to pull the reference data. You will also need it if you wish to re-analyze older seizures. However, if the lab team has sent you the new seizure as a stand-alone file, which is their usual approach, and you already have pre-processed reference and a familial matching archive, you can skip section 3 of this documentation and do not need the master elephant database.

*Resources and their locations*

The pipeline programs need to marshall a large number of resources, such as programs and scripts, maps of Africa, databases of reference samples, and prior familial matching results. Locations of these resources are described in the file "ivory\_paths.tsv". A sample is provided in ivory\_pipeline/aux/ but you will need to adjust it to your own system. The good news is that once this is done, subsequent runs are much more straightforward.

The ivory\_paths.tsv file is tab-separated (no spaces!) and all pathnames must end with "/". In general the first entry is the name of the resource being located, the second is its path, and the third, if present, is the name of the resource. This documentation will be easier to read if you follow along in the sample ivory\_paths.tsv file found in ivory\_pipeline/aux/.

a) *Location of the ivory\_pipeline directory.*  The resource name is "ivory\_pipeline\_dir" and the second entry is the full path to where you have placed the ivory\_pipeline directory. Directions for obtaining the ivory\_pipeline directory from GitHub are given below. This documentation is found in ivory\_pipeline/docs and if you obtained a copy from somewhere else, you should check there for the most up-to-date version.

b) *Zone files.* These files relate the sampling zone (e.g. Kibale National Park) to its ID number and location (decimal latitude and longitude). They need to be updated when samples are received from new areas and new zones are created to hold them: this is currently done by Ada. There are two zone files, one for forest and one for savannah; zones which contain both species are present in both files with the same ID number. Current zone files are in ivory\_pipeline/aux/ but check with Ada for more recent ones.

The resource name is "zones\_prefix". The second entry is the full path to where these files are found, and the third is the prefix of the files. For example, if you plan to use zones\_43\_forest.txt and zones\_43\_savannah.txt, you would put "zones\_43" here.

c) *Map files.* These files show the range in which each species may be found. We normally use the versions starting with "mapfile\_161220", which were created by Sam and Mary and are permissive--areas are included if there's any chance there are or were elephants there. (We also have more stringent versions, "iucn\_161220", based on IUCN range maps. These have not been well tested; it is possible SCAT cannot search them adequately. I recommend use of the "mapfile\_161220" versions.) For each map there is a forest version, for example mapfile\_161220\_forest.txt, and a corresponding savannah version. Areas with both species potentially present are included in both maps. Current mapfiles are in ivory\_pipeline/aux/.

The resource name is "map\_prefix". The second entry is the full path to where these files are found, and the third is the prefix of the files, for example "mapfile\_161220".

d) *SCAT executable.* We provide a copy of the SCAT code with pipeline-specific modifications. You will need to compile it (described below) and set this line to point to the location of the executable, which will generally be ivory\_pipeline/src/scat-master/src. The first entry is "scat\_executable", the second is the full path to this executable, and the third is the executable's name (typically "SCAT3").

e) *VORONOI executable.* We provide a copy of the VORONOI code, modified to work with the pipeline, in ivory\_pipeline/src/voronoi-master/src/. You will need to compile it (described below) and then set this line to point to the location of the executable. The first entry is "voronoi\_executable", the second is the full path to this executable, and the third is the executable's name (typically "VORONOI").

f) *STRUCTURE executable.* We provide a copy of the STRUCTURE code with minor changes to make it compile on recent compilers. You will need to compile it (described below) and then set this line to point to the location of the executable, which will generally be in ivory\_pipeline/src/structure\_kernel\_src. The first entry will be "structure\_executable", the second the full path to this executable, and the third is the executable's name (typically "structure").

f) *Reference data files.* If you have obtained pre-generated reference files, you will set up a directory named after the reference version (for example REFELE\_5.8) and this entry will point to the *parent* of that directory. If you are making your own reference files by following the instructions in section 3, you will create this directory during the section 3 process, and again the ivory\_paths.tsv entry will point to its parent. The first entry is "reference\_prefix", the second is the path to the parent directory in which the reference directory is found, and the third is the reference prefix. For example, my reference data are in /home/mkkuhner/data/REFELE\_5.8/, so my second entry would be "/home/mkkuhner/data/" and my third would be "REFELE\_5.8".

g) *Seizure data files.*  Establish a directory in which you will put raw seizure data files. The first entry is "seizure\_data\_dir" and the second is the full path to this directory.

The following resources are needed only if you plan to run familial matching.

h) *Familial matching archive.* This archive contains familial matching information from all previously analyzed seizures. It is not kept on GitHub for data security reasons. It is possible to regenerate this file from the master elephant database, but this takes several days as familial matching has to be run on all 75+ seizures. Note that the archive directory name contains the reference prefix against which it was run, and you can only use it if you are using the same reference; if the reference data change, you will need to rerun everything.

Your results will be added to this database, so if you received it from someone, be sure to send a copy back to them so that future seizures can be matched against yours.

The first entry is "fammatch\_archive\_dir". The second is the path to the *parent* directory of familial matching archives. The third is the path to the specific version you are using. For example, I keep my familial matching archive on an external hard drive. All versions are in "/media/mkkuhner/ElephantDataArchives/elephant\_archive/" which will be the second entry on this line. The version for reference 5.8 is then in a subdirectory "elephant\_archive\_REFELE\_5.8/" and that will be the third entry in this line.

g) *Seizure modifications file.* By default, the final familial matching reports will include all seizures. However, our full "seizure" list include things which you may want to omit or merge together. For example, certain seizures from the Philippines were apparently mixed together in storage, and matches between them are likely due to this mixing and not common origin: we typically merge these. The ivory\_pipeline/aux/ directory has a file "seizure\_modifications\_example" which gives the modifications in effect for our Nature paper. Instructions for making this file are in section 4D. In ivory\_paths.tsv the first entry is "seizure\_modifications\_prefix", the second is the path to the file, and the third is the file's name. The seizure\_modifications file does not otherwise affect what the pipeline does: seizures mentioned in it are processed entirely normally up until the final familial-matching reports.

h) *Additional files for the familial matching final report.* To create an overall familial matching report, you will need a file of exact matches, **dms.tsv**. This is not public due to containing seizure information. Mary or Ada can provide a copy. It does not appear in ivory\_paths.tsv (WHY NOT?) You will also need a file of simulation results establishing false-positive rates, **fprates.tsv**, which is found in ivory\_pipeline/aux.

Creating network graphs based on familial matching results will need a file **port\_colors.tsv** giving the correspondence between colors and ports. A sample is in ivory\_pipeline/aux but if your data have additional ports you will need to add them. You will also need a file **seizure\_numbering.tsv** relating the seizure names to seizure numbers (the network graphs use numbers as the names don't fit) as well as giving the port of each seizure. Note that the country in the seizure name is the seizing country, whereas port refers to the exporting country. A sample is in ivory\_pipeline/aux/seizure\_numbering\_20220407.tsv but will need to be updated with any seizures you have added.

The following table summarizes the contents of "ivory\_paths.tsv".

|  |  |  |  |
| --- | --- | --- | --- |
| tag | information1 | information2 | needed for |
| ivory\_pipeline\_dir | path to "ivory\_pipeline" directory |  | everything |
| scat\_executable | path to directory of SCAT executable | filename of SCAT executable | species assignment, SCAT |
| voronoi\_executable | path to directory of VORONOI executable | filename of VORONOI executable | VORONOI |
| structure\_executable | path to the directory containing STRUCTURE executable | filename of the structure executable | species ID |
| reference\_prefix | path to location of reference data | reference prefix (e.g. REFELE\_5.8) | everything |
| zones\_prefix | path to zonefiles | prefix of zonefiles (e.g. zones\_43) | species assignment, SCAT |
| metadata\_prefix | path to the metadata file associated with the familial matching archive (this file links SIDs to seizures) | metadata filename (currently seizure\_metadata) | familial matching |
| seizure\_modification\_prefix | path to the seizure modifications file | filename of the seizure modifications file | familial matching |
| map\_prefix | path to the map files | prefix of the map files (e.g. mapfile\_161220) | SCAT, VORONOI, species ID |
| seizure\_data\_dir | path to the directory containing raw seizure data |  | species ID |
| fammatch\_archive\_dir | directory of fammatch archive | subdirectory for this reference database (e.g. elephant\_archive\_REFELE\_5.8/) | familial matching |

2B. Code and libraries

The following table gives the original GitHub site for programs used in this pipeline. However, except for the ivory pipeline itself, the GitHub versions are mostly unsuitable for pipeline runs, and I recommend using the versions packaged with the pipeline. For example, the GitHub STRUCTURE is very old code and recent gcc refuses to compile it; the GitHub VORONOI has highly inconvenient input and output which the pipeline cannot handle, whereas this has been rewritten in the included version.

|  |  |  |  |
| --- | --- | --- | --- |
| Program | Location within ivory\_pipeline bundle | GitHub repository | Language |
| Ivory pipeline | -- | https://github.com/mkkuhner/ivory\_pipeline | Python 3 |
| EBhybrids | /src/EBhybrids | https://github.com/stephenslab/EBhybrids | R |
| Fammatch | /src/elephant\_fam\_match | https://github.com/cwolock/elephant\_fam\_match | R |
| SCAT | ivory\_pipeline/src/scat-master | https://github.com/stephens999/scat | C++ |
| VORONOI | ivory\_pipeline/src/voronoi-master | https://github.com/stephens999/voronoi | C++ |
| STRUCTURE | ivory\_pipeline/src/structure\_kernal\_src | https://web.stanford.edu/group/pritchardlab/structure\_software/release\_versions/v2.3.4/html/structure.html | C |

To obtain the ivory\_pipeline code itself, install "git" on your system and then type:

git clone https://github.com/mkkuhner/ivory\_pipeline

The C and C++ programs will have to be compiled on your system. We are able to do this with gcc/g++ using the included Makefiles. SCAT uses the LAPACK library: on Ubuntu this can be installed with "sudo apt install liblapack-dev". (It is a FORTRAN library and may require having a FORTRAN compiler available.)

*Python scripts.* Many scripts call routines from our project library ivory\_lib.py, which must be present in the same directory as the script (not necessarily the directory where you are running it, but the directory in which the script itself is located). So if you copy Python scripts into a new directory, be sure to copy ivory\_lib.py too.

Various scripts in the pipeline use packages which are not part of the default Python3 installation. These can often be installed with "pip3 install <packagename>" or, on Ubuntu, "sudo apt-get install <packagename>" (see the following table for package names for use with apt-get). NB: On recent versions of Ubuntu, pip3 will only install into a virtual environment, and you will then have to run the pipeline in that virtual environment. There is no apt-get install option for the *haversine* library, so your choices are either running in a virtual environment, or forcing a pip3 install which we're warned may break your system Python installation--though it did not break mine.

|  |  |  |
| --- | --- | --- |
| **Library** | **Purpose** | **Ubuntu package name** |
| haversine | measure distances on the globe | N/A |
| numpy | statistics routines and data structures | python3-numpy |
| matplotlib | plotting | python3-matplotlib |
| cartopy | maps and coordinate conversions | python3-cartopy |
| pycairo | network graphics | libcairo2-dev pkg-config python3-dev |
| networkx | network construction | python3-networkx |
| community | network construction | python3-louvain |
| graph\_tool | network graphics | python3-graph-tool |

**3. Processing reference data**

This entire section can be skipped if you have processed reference files and a familial matching archive already available. Use it if you need to build new ones, for example because a new release of the reference data has become available. Note that a familial matching archive from a previous release of the reference files will *not* work with a new release: every familial matching run must use exactly the same reference or they cannot be combined.

**3A. Extracting data from master database**

The master elephant database will be an .xlsx file, for example Elephant\_Genotypes\_Master\_5.8.xlsx; the corresponding REFPREFIX would be REFELE\_5.8. Open this file and save the "Reference Genotypes Stats" tab as a .csv file named REFPREFIX\_raw.csv. Be sure you do not use the "Reference Genotypes All" tab, which includes samples that failed in genotyping. Place this file in the location given in your ivory\_paths.tsv file in the entry "reference\_prefix".

Also save the "Ivory Genotypes Stats (10+ loci) tab as REFPREFIX\_seizures.tsv in the same location.

**3B. Preparing data files and familial matching archive**

For this step you will run the pipeline script step0\_setup.py as follows:

python3 step0\_setup.py REFPREFIX REFPREFIX\_seizures.tsv ivory\_paths.tsv

This script performs three tasks. It prepares the reference data for use; makes raw seizure data files for all seizures mentioned in the database (though does not process them further); and creates a new familial matching archive.

Preparing the reference data requires species identification and hybrid removal. The use of EBhybrids to identify species relies on having some individuals of known species. We run Structure with the number of populations set to two, and pick out the most clearcut individuals of the two populations to serve as a reference for EBhybrids. To decide which population is which, we look for elephant CH0878 (from Chobe National Park, Botswana) and assume the population containing it is savannah. If you have a reference data set not containing CH0878, you will need to substitute a different elephant in Python script make\_eb\_input.py, found in ivory\_pipeline/src.

The Structure program relies on two parameter files "mainparams" and "extraparams" which need to be in the same directory as the Structure executable (which is where Structure is going to run). The reference pipeline *overwrites* the copy of mainparams in the Structure directory with a version that points to your desired input and output files and has the correct count of samples. If you want to retain the original values, rename or move away the original mainparams before running this pipeline.

The step0\_setup script runs Structure, which will dump text to the screen for several minutes: this is normal. It then runs EBhybrids for species ID and hybrid detection. When it's done, needed reference files will be in subdirectory REFPREFIX off the directory where you ran it. In particular, REFPREFIX\_filtered\_forest.txt will contain your forest reference samples, culled of hybrids, in SCAT format; and REFPREFIX\_filtered\_savannah.txt will contain your savannah samples. Make sure that your ivory\_paths.tsv file points to this directory as its source of reference data. Is the "filtered" in here really correct for downstream code?

It will also place raw seizure files in the location noted in ivory\_paths.tsv for seizure data, and an empty familial matching archive in the location noted in ivory\_paths.tsv for that.

*Warning note on EBhybrids:* This program is stochastic, and if you run it twice, a borderline elephant may move from one call to another (in practice, between hybrid and one of the pure species: no known samples are ambiguous between the pure species). The ivory pipeline therefore always uses the reference EBhybrids results from the original reference run. Otherwise different seizures will use slightly different reference data, with bad effects downstream. This means that if someone hands you a reference data file without its EBhybrids files, (a) it won't run until you recreate those files, and (b) you may not get exactly the same results they did. There is one elephant whose hybrid probability is almost exactly 0.5, and it will be a hybrid in some runs and not in others even with identical inputs.

**4. Processing seizure data**

**4A. Obtaining the seizure data**

For a new seizure, the lab team typically sends an .xlsx file with a tab "For SCAT". Save the contents of this tab with the filename PREFIX\_raw.tsv (tab separated) and place it in the directory you've indicated as the location of seizure data files (in your ivory\_paths.tsv file).

If you are re-running an older seizure, you can extract seizure data from the master elephant database .xlsx file (Ada is the maintainer of this file). To do this, save the "Ivory Genotypes Stats (10+ loci)" tab from the master database as a .tsv file. Provide the name of this file to Python script make\_raw\_seizure.py as follows:

python3 make\_raw\_seizure.py databasefile.tsv ivory\_paths.tsv

This script will construct raw data files for *all* of the seizures in the master database and place them in the directory designated by your ivory\_paths.tsv file as the location of seizure data files.

**4B. Data preparation: Validation, species identification, and hybrid removal**

In "step 1" the pipeline validates the data, identifies the species and hybrid status of each elephant, reports on and removes hybrids, and separates the remainder by species. This can either be run on its own (step1\_nofam.py) or combined with familial matching (step1\_fammatch.py).

In either case, this program should be run in the parent directory of all seizures. It creates a seizure directory named with the seizure PREFIX and will refuse to run if that directory already exists, to prevent unwanted overwriting: delete, rename, or move the old directory if necessary.

The key input data for this program is PREFIX\_raw.tsv, which the script expects to find in the seizure data directory specificed in the ivory\_paths.tsv file.

To run with familial matching:

python3 step1\_fammatch.py PREFIX ivory\_paths.tsv

To run without familial matching:

python3 step1\_nofam.py PREFIX ivory\_paths.tsv

*Validation steps:* The program checks that needed directories and files exist. It logs the run in file PREFIX/PREFIX\_logfile.txt, recording what version of each program is being used; this can be helpful in documenting the results.

It then runs verifymsat.py, which checks the msat names and looks for unexpected alleles. *The program may fail at this point because the lab team can't decide if the first entry in the header is MatchID or SampleID.* (Can we fix that?) My code assumes it is MatchID; hand-correct if not. The program will report on unexpected alleles--ones never observed for that locus in the reference database. A small number (zero to two per locus) is normal. A larger number suggests the msats are out of order, uncalibrated, or otherwise problematic, and will cause the pipeline to stop. Don't go any further until this is resolved. (For example, Julie Bonnald's data hung up here because her canonical msat order was different from ours. If such a run had been forced to continue the results would be nonsense.)

It runs prep\_scat\_data to remove individuals with too much missing data, and then detect\_duplicates to look for within-seizure exact matches. If any are found they should be reported to the database maintainer so that their entries can be consolidated, and pipeline execution should be put on hold until this is done. Including duplicate entries within a seizure biases VORONOI, and using incorrect MatchIDs (since the MatchIDs of duplicates will be changed) messes up everything. Start over with step 1 once you receive corrected input data.

The program then goes on to run EBhybrids for hybrid detection and species identification, using the EBhybrids R scripts found in ivory\_pipeline/src. This writes a report called PREFIX/PREFILE\_hybout.txt and also preserves the raw EBhybrids output files as PREFIX\_hybt.txt and PREFIX\_HPs.txt (and .csv versions of those). The lab group should be sent a copy of PREFIX\_hybout.txt so that they can update the database.

Finally, it creates species-specific input directories (subdirectories of PREFIX), species-specific seizure data files, and species-specific run command files for SCAT. It will create a directory "PREFIX/nsavannah" if any savannah samples were found, and a directory "PREFIX/nforest" if any forest samples were found. The pipeline will likely come to a sudden halt here if all samples were hybrids; nothing downstream can be run in that case as we can neither geolocate hybrids nor assess their familial matching, due to inability to define the reference population. No current seizures are all hybrids.

**4C. Familial matching**

If the step1\_fammatch.py script is running, it will then go on to do familial matching, separately for each species present. It begins by using SCAT to assign samples to sectors (e.g. "southern savannah"). Only a single and rather quick SCAT run is used here, and it is directly run by the script. It then preps the data and runs the fammatch R scripts using the versions in /ivory\_pipeline/src. This step requires access to the fammatch database. Note that no reports are generated here, because familial matching probabilities have to be assessed in the context of the entire set of seizures, so it is useless to run them until all seizures are done. This global evaluation will be done in step4\_finalreports.py.

If you initially ran step1\_nofam.py and now want to go back and add familial matching, you cannot just run step1\_fammatch.py as it will detect that the seizure directory already exists and refuse to proceed. I recommend using step1\_fammatch.py unless you are certain familial matching is inappropriate, because otherwise you'll have to run individual scripts by hand here, or else start over from scratch. TO DO: split the fammatch and non-fammatch process in such a way that fammatch can be added gracefully later.

**4D. SCAT**

SCAT geolocates each sample in a seizure individually--all as one run, but the samples do not influence each others' placement. SCAT runs on a large seizure can take a day or two, and we need nine of them. As a result, the script step2\_scat.py sets up everything for the SCAT runs, but does not actually start them: they must be started by hand. On a machine with multiple processors, or a cluster, it is very helpful to start more than one at a time; they are run in separate directories and do not interfere with each other.

If the argument "cluster" is passed to step2\_scat.py, it will set up the SCAT run command files to work on the cluster. Note that step2\_scat.py itself is not designed to be run on the cluster (this would be pointless as it runs in a couple of seconds): instead, you will copy the directories it generates, which include run command files, onto the cluster and then execute the SCAT runs there. RYAN: Can we make the "cluster" option more general? Right now I fear it will make files that only work on my account. It should also write a file that allows launching of all 9 or 18 runs with a single command.

Need to document needed Docker containers here for cluster runs.

To run step2\_scat.py use the following command (run in the parent directory of all seizures):

python3 step2\_scat.py PREFIX laptop/cluster ivory\_paths.tsv

Once step2\_scat.py is finished you will need to execute the nine run commands (or eighteen if both species are present in the seizure). On Ubuntu I do this as follows:

Open a new window. Navigate to the PREFIX directory. Note whether directories "nsavannah" and "nforest" are present. For each one that is present, change directory into it. You will see nine directories named 1, 2 ... 9. Change directory into 1 and you will find a file named run1.sh. Execute this file with "source run1.sh". You should see a bit of immediate output. Now open a new window and repeat with 2, 3...9, executing run2.sh, run3.sh, ... run9.sh. If your machine can't handle running all of them at once, run a few at a time. On Ubuntu I use the "top" command to look at CPU utilization. In general you should reserve one processor for system operations; if you fill up all the processors everything will slow down. On a large seizure (200+ samples) each of these runs may take up to a couple of days. Having each one in its own window allows you to see how they're doing and whether they have finished or crashed.

The code which produces a table of point estimates and heatmaps for each elephant is normally run after VORONOI. If you are not running VORONOI you will want to run that code by hand now. It is program plot\_results.py in /ivory\_pipeline/src and should be run in the parent directory of all seizures (not in the PREFIX directory, but one step above). It will diagnose absence of VORONOI results and behave appropriately. To run it:

python3 plot\_results.py PREFIX ivory\_paths.tsv

Reports and images will be found in the PREFIX/reports directory.

**4E. VORONOI**

The VORONOI program post-processes SCAT output to give improved elephant locations on the assumption that the elephants in the seizure are clustered relative to elephants of their species across Africa. If this assumption is not sound, VORONOI will be misleading. For example, we ran SCAT on an ivory chess set made in Hong Kong. We found a large number of different elephants, both forest and savannah. We suspect this chess set was made from scraps from a large ivory-working shop and there is no reason to think the elephants came from the same location, so VORONOI would not be appropriate. VORONOI also should not be run on seizures containing fewer than 10 samples, as it does not seem to add value; the automated pipeline will skip VORONOI unless there are at least 10 samples.

Caveat about VORONOI: This code is known to have bugs which cause its search to blur out somewhat rather than focusing on the optimum. Paradoxically, its tested performance is better with the bugs present than with them fixed. I hope to replace VORONOI completely with CLUST once that program is fully debugged and tested; in the meantime I'm afraid it's best to run the buggy version.

VORONOI must be run after all SCAT runs have completed. It can take up to several hours for a large seizure. Only one run is needed. To run it:

python3 step3\_voronoi.py PREFIX ivory\_paths.tsv

The step3\_voronoi.py program ends by creating a file of point estimates and a set of heatmaps showing the inferred SCAT and VORONOI results. These will be found in PREFIX/reports. Note that heatmap colors are relative. An elephant which is poorly localized will have some yellow and white pixels but these may represent extremely weak support for that location--just less weak than other locations.

**4F. Report generation**

*Basic reports.*Reports onhybrids (PREFIX\_hybout.txt) and on SCAT and VORONOI results (directory PREFIX\_reports) have already been generated. Program step4\_finalreports.py draws conclusions about familial matching based on information from all seizures. Its results will be in a directory off the parent directory of all seizures named fammatch/overall.

Our current statistical approach to dealing with the high number of false positives in familial matching requires estimating the false-positive rate for various likelihood ratio values based on the entire seizure data set. Thus, step4\_finalreports.py should be run only when all desired seizures have completed familial matching (step1\_fammatch.py). It does not rely on SCAT or VORONOI results.

The fammatch program can only detect exact matches when they are perfect (aside from missing data). However, we believe that pairs of samples which differ only by a mismatch of the form AA vs AB are overwhelmingly likely to be exact matches with allelic dropout, and that may also be true for 2 or 3 mismatches of this kind. CERVUS (Kalinowski 2007) can be used to verify such conclusions. (CERVUS is not included in the ivory pipeline, as it's a Windows program, and is normally run by Ada.)

To handle exact matches not detectable by fammatch, we maintain a file of all known cross-seizure exact matches (file called "dms.tsv" based on the earlier term "direct matches"). This file is not public due to containing seizure information; you will need to obtain a copy from Mary or Ada. Its format is SID1, SID2, seizure1, seizure2, species (S=savannah, F=forest). The one known hybrid exact match is NOT included here since familial matching is not done on hybrids, so its inclusion would bias the results. Be sure to update this file if additional cross-seizure exact matches are found.

The post-analysis of fammatch also requires a file of false-positive rates derived from simulated data, called fprates.tsv and found in ivory\_pipeline/aux.

NOTE: Users may press to have a list of individual matches. False positives are frequent except at very high likelihood ratio. I strongly prefer to give the seizure-to-seizure matching score generated by step4\_finalreports.py, and not the individual matches (unless they are exact matches), as they are highly susceptible to overinterpretation.

This program needs a likelihood ratio cutoff value, below which putative matches are deemed too weak to count at all. The simulation data are set up with a cutoff of 2.0, and I am not positive other cutoffs will run successfully. In any case it should not, in my opinion, ever be set lower than 2.0. It also needs a minimum number of loci that must be successfully typed in both individuals in order to allow that pair to be considered (note that this is different from the minimum loci for an individual to be valid). Wasser et al. 2022 used 10.

This program should be run from the parent directory of all seizures, as follows:

python3 step4\_finalreports.py ivory\_paths.tsv dms.tsv LR\_cutoff min\_loci

Its reports will be found in a directory off the parent directory of all seizures, called fammatch\_overall. The file sig\_seizure\_edges.csv in this directory gives all seizure pairs with an expected number of matches >= 1 and is useful if, for example, you just want to report which seizures are connected to your current one.

*Network diagrams.* The two files seizure\_nodes.csv and seizure\_edges.csv are the raw input for clustering seizures and creating network diagrams (Wasser et al. 2022). The step4\_finalreports.py script does not produce the network diagrams, because getting them to be readable requires manual intervention during the run--you need to move nodes around so that they are not on top of each other and the lines between them can be seen. You will then save that configuration so you don't have to do it again, and can experiment with different colors or labels without redoing the untangling process. Thus, you are very likely to need to run this multiple times. BEWARE: if you don't save the configuration, you will never be able to get it exactly the same, which is a big problem in producing figures that show the same network with different labels or colors.

This is the worst part of the pipeline because it is seldom used, complex, and arbitrary. I'm sorry about the state it's in!

To make network diagrams you will need several auxillary files. Seizures are numbered rather than named, because there isn't room for the names. The numbering key is a tab-separated file of seizure name to number: an example "seizure\_numbering\_20220407.tsv", used for the NatCom paper, is in ivory\_pipeline/aux. Any new seizures will have to be added here, and any renamed seizures will have to be corrected.

You will also need a file giving the color to use for each port of origin. An example (color blindness friendly) used for Wasser et al. 2022 is in "port\_colors.tsv" in ivory\_pipeline/aux. The program expects to find this exact filename in this directory, rather than taking it as an argument. If your data include ports not mentioned here you will need to pick colors for them: the last color in the example file seems to be used for "other" so might be an appropriate choice.

To make network diagrams for the first time on a given data set, run the following:

python3 draw\_network.py minlink ivory\_paths.tsv seizure\_numbering.tsv None

This will write a file layout.pkl. For subsequent runs where you want to use the same layout of the network nodes, instead run it as follows:

python3 draw\_network.py minlink ivory\_paths.tsv seizure\_numbering.tsv layout.pkl

TO DO: try this and see if it actually runs! (I doubt it.)

5. Future directions

*Microsatellite data.* Currently we use estimates of dropout rates per microsatellite locus made for Mondol et al. (2015). These are very old now, but I have not been able to re-estimate them as I can't get Mondol's scripts (the two scripts beginning with "infer" in the EBhybrids directory) to run. The estimates we are using, found in file ivory\_pipeline/aux/dropoutrates\_savannahfirst.txt, seem rather high and it would be good to update them. We should not use estimates of our current lab dropout rate, however, as much of our data was genotyped years ago.

*SNP data.*  SCAT, VORONOI, and STRUCTURE will run unmodified. New code will be needed to make SNP input files for SCAT and STRUCTURE. EBhybrids and fammatch assume microsats and will need versions that can handle SNPs. The simulated data used in final fammatch reports to estimate false-positive rates will have to be redone using SNPs.

*Pangolin SNP data.* The SNP changes mentioned above will be required. Additionally, EBhybrids only works for two species. STRUCTURE is a potential replacement, especially if hybrid pangolins do not occur (the literature suggests they don't). I recommend running EBhybrids on pairs of species to verify this. New mapfiles will be needed to indicate the four species ranges, and new simulations for the final fammatch report will be needed: we can't assume that elephant SNPs and pangolin SNPs give the same false-positive rates in fammatch. Finally, the code loops over forest/savannah in many places and all of these will need to be modified to handle the four pangolin species.

6. References

*EBhybrids:* Mondol, S., Moltke, I., Hart, J., Keigwin, M., Brown, L., Stephens, M. and Wasser, S.K. (2015), New evidence for hybrid zones of forest and savanna elephants in Central and West Africa. Mol Ecol, 24: 6134-6147. <https://doi.org/10.1111/mec.13472>

*Structure:* Pritchard JK, Stephens M, Donnelly P (2000) Inference of population structure using multilocus genotype data. *Genetics*, 155, 945–959.

*SCAT*: Wasser SK, Shedlock AM, Comstock K, Ostrander EA, Mutayoba B, Stephens M. Assigning African elephant DNA to geographic region of origin: applications to the ivory trade. Proc Natl Acad Sci U S A. 2004 Oct 12;101(41):14847-52. doi: 10.1073/pnas.0403170101.

*VORONOI:* Wasser, S.K., Mailand, C., Booth, R., Mutayoba, B., Kisamo, E., Clark, B., & Stephens, M. (2007). Using DNA to track the origin of the largest ivory seizure since the 1989 trade ban. Proceedings of the National Academy of Sciences, 104(10), 4228-4233.

*Fammatch:* Wasser, S.K., Wolock, C.J., Kuhner, M.K. *et al.* Elephant genotypes reveal the size and connectivity of transnational ivory traffickers. *Nat Hum Behav* **6**, 371–382 (2022). https://doi.org/10.1038/s41562-021-01267-6

*CERVUS:* Kalinowski, S.T., Taper, M.L., & Marshall, T. C. Revising how the computer program CERVUS accommodates genotyping error increases success in paternity assignment. Mol. Ecol. 16: 1099-1106. (2007).