Ancestry Prediction in scRNAseq

Release 0.0.3

Drew Neavin

Nov 22, 2022

CONTENTS

CHAPTER

ONE

BACKGROUND

1.1 Motivation for this Pipeline

Genetic ancestry is an important piece of information when assessing genetic and transcriptomic data across multiple different individuals.

Often, single cell RNA-sequencing data is produced on samples from donors whose personal information is unknown to the researchers. For example, ancestral inforamtion is often not ascertained at sample collection and self-reporting can sometimes be misleading. Single nucleotide polymorphism (SNP) genotyping of excess sample can be used to estimate ancestry. However, excess sample is not always available - especially when using publicly available data. Of course,

With this in mind, we established this pipeline to estimate genetic ancestry from scRNA-seq data by:

1. Estimating genetic information from the scRNA-seq reads (using freebayes)
2. Aligning the samples to 1000 Genomes principal component (PC) space
3. Predicting the ancestry by training a k nearest neighbors model on the 1000 Genomes data.

Note

We have provided instructions to run this pipeline in two ways:

1. Through a *snakemake pipeline* (suggested especially for datasets with multiple pools and individuals)
2. *Manually* which can be used when data do not fit the assumptions in the snakemake pipeline or just to get a better idea of the steps that are implemented in the snakemake pipeline.

1.2 First Steps

First, proceed to the *Installation* section to download the required singularity image and set up the pipeline locally.

1.3 Support

If you have any questions, suggestions or issues with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue or email Drew Neavin (d.neavin @ garvan.org.au)

CHAPTER

TWO

INSTALLATION

This pipeline to estimate genetic ancestry from scRNA-seq data has been built in snakemake and packaged in a Singularity image that contains all the required softwares. This should provide continuity to execution of this pipeline across different computational systems. We're hoping that installation and pipeline execution will be relatively painless as well.

2.1 Singularity Image

The only thing to note before you download this image is that the image is **~6.5Gb** so, depending on the internet speed, it will take **~15-30 min to download**.

To download the singularity image:

```
wget https://www.dropbox.com/s/nnru32uw1ate2gu/ancestry_prediction_scRNAseq.sif
wget https://www.dropbox.com/s/pz3h9gzhcn2kgob/ancestry_prediction_scRNAseq.sif.md5
```

Then you should check to make sure that the image downloaded completely by comparing the image md5sum to the original md5sum. You can do that by running the following commands:

```
md5sum ancestry_prediction_scRNAseq.sif > downloaded_ancestry_prediction_scRNAseq.sif.md5
diff -s downloaded_ancestry_prediction_scRNAseq.sif.md5 ancestry_prediction_scRNAseq.sif.
↪md5
```

If everything was downloaded correctly, that command should report:

```
Files ancestry_prediction_scRNAseq.sif.md5 and downloaded_ancestry_prediction_scRNAseq.
↪sif.md5 are identical
```

Note

Please note that the singularity image and this documentation is updated with each release. This means that the most recent documentation may not be 100% compatible with the singularity image that you have.

You can check the version of your singularity image to match with documentation with:

```
singularity inspect ancestry_prediction_scRNAseq.sif
```

If you run into any issues with downloading the image or any issue with running anything from this image, you can reach out to us by submitting an issue at Github

Software versions - for the curious

Image build date: 24 July, 2022

Software Group	Software	Version
Supporting Softwares	sinto	0.8.4
	Crossmap	0.6.4
	vartrix	v1.1.3
	htslib	v1.13
	samtools	v1.13
	bcftools	v1.13
	freebayes	v1.3.5
R Supporting Packages (R v4.2.1)	argparse	v2.1.6
	ComplexHeatmap	v2.12.0
	data.table	v1.14.2
	vcfR	v1.13.0
	tidyverse	v1.3.2
	cowplot	v1.1.1
	colorspace	v2.0-3
	ggplot2	v3.3.6
	caret	v6.0-92
	RColorBrewer	v1.1-3
Python Supporting Packages (Python v3.6.8)	argparse	v1.4.0
	pysam	v0.19.1
	pandas	v1.1.5
	scipy	v1.5.4

2.2 Next Steps

The next section *Data Preparation* will explain the input files required for this software and their expected formats.

2.3 Support

If you have any questions, suggestions or issues with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue or email Drew Neavin (d.neavin @ garvan.org.au)

CHAPTER

THREE

DATA PREPARATION

This section will explain the required inputs for this ancestry prediction pipeline from scRNA-seq data and the expected formats for those files.

3.1 Data Files

Here is a list of the files that you will need for this pipeline with further explanation of preparation of the files below:

Required

- *Bam file(s)* - The single cell bam files
- *hg19 reference fasta file*
- *Annotated barcodes file* - Two-column tab separated file indicating which barcodes match to each individual in the dataset
- *Sample metadata file* - Two-column tab separated file that contains the names of the pools and the IDs of the individual in each pool

Optional

- *hg38 reference fasta file* - ONLY NEEDED IF YOUR SEQUENCE DATA WAS MAPPED TO HG38!!!
- *Reference vcf file* - Only needed if you have reference SNP genotypes from a microarray or called from whole genome or exome sequencing data and want to compare the ancestries predicted from the reference SNP genotypes to the ancestries predicted from the scRNA-seq data.

3.2 Preparation of Data Files

3.2.1 Bam File(s)

Required

Bam file(s) that contain the aligned sequences from the scRNA-seq.

3.2.2 hg19 Reference Fasta File

Required

A reference hg19 fasta that uses the same chr encoding as you bam file (i.e. chromosomes are encoded as ch1 or 1)

3.2.3 Annotated Barcodes Files

Required

A tab-separated file that has the barcodes in the first column and the IDs of the individual that they are assigned to in the second column. This file should NOT have a header.

Note

Please make sure that your individual IDs do not start with a number as there are some softwares in the pipeline that do not handle them well.

For example:

AAACCCAAGAACTGAT-1	K835-8
AAACCCAAGAAGCCTG-1	K1292-4
AAACCCAAGCAGGTCA-1	K1039-4
AAACCCAAGCGGATCA-1	K962-0
AAACCCAAGCTGCGAA-1	K835-8
AAACCCAAGGTACTGG-1	K1292-4
AAACCCAAGTCTTCCC-1	K835-8
AAACCCACAACCGCCA-1	K835-8
AAACCCACACAGTGAG-1	K962-0
AAACCCACACCCTGAG-1	K835-8
…	…

3.2.4 Sample Metadata File

Required

A tab separated file that contains two columns: the first for the Pool and the second for the

---

Note

Please make sure that your individual IDs do not start with a number as there are some softwares in the pipeline that do not handle them well.

Pool	Individual
RZ731_Pool8	K1039-4
RZ731_Pool8	K1292-4
RZ731_Pool8	K752-4
RZ731_Pool8	K835-8
RZ731_Pool8	K938-0
RZ731_Pool8	K962-0

3.2.5 Reference SNP Genotypes vcf

Optional

If you have reference SNP genotypes for the individuals in your dataset from microarray or whole exome or genome sequencing, we have build funcionality into the pipeline to estimate ancestry based on the referene genotypes and provide comparison between the reference and scRNA-seq predicted ancestry annotations.

3.2.6 hg38 Reference Fasta File

Optional

ONLY NEEDED IF YOUR SEQUENCE DATA WAS MAPPED TO HG38!!!

3.3 Support

If you have any questions, suggestions or issues with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue or email Drew Neavin (d.neavin @ garvan.org.au)

CHAPTER

FOUR

PIPELINE EXECUTION

We have provided a complete pipeline to execute ancestry prediction from single cell data that has been built with Snakemake and packaged into a Singularity image with all the required softwares and supporting files.

To run your dataset(s) manually, procded to the *Preparing to Execute the Pipeline* documentation.

4.1 Preparing to Execute the Pipeline

The files required to execute the snakemake pipeline (except those listed in the *Data Preparation section*) are packaged into the Singularity image. Some of those files need to be transfered to the local system for pipeline execution. The following sections help transfer these files to your local system and provide instructions on how to edit thtem

4.1.1 Image Setup

Required

To get files from the Singularity image onto your system, simply execute:

```
singularity run --bind <absolute_directory_path> --app setup ancestry_prediction_
↪scRNAseq.sif <absolute_directory_path>
```

Note

The pipeline expects certain files pulled from the image to be in the same directory as the singularity image so you will have to rerun the setup steps if you move the image

This will transfer some of the files from the Singularity image to your local system:

```
.
├── ancestry_prediction_scRNAseq.yaml
├── includes
│   ├── reference_ancestry_predictions.smk
│   └── souporcell_ancestry.smk
├── mods
│   └── prepareArguments.py
├── Snakefile
└── snakemake.yaml
```

We have included a yaml (`snakemake.yaml`) to generate a conda environment that has snakemake and accompanying software required to execute the pipeline as well as the `ancestry_prediction_scRNAseq.yaml` which will need to be edited based on your files and system.

4.1.2 Install Snakemake

Required

You will need snakemake to run the pipeline. We highly recommend using the conda environment that we have generated as it has all the required dependencies. You can create this environment on your system using the `snakemake.yaml` that we have provided.

```
conda env create -f snakemake.yaml -n ancestry_pred_snakemake
```

Then, to activate this environment:

```
conda activate ancestry_pred_snakemake
```

If you would prefer to install snakemake and scipy yourself, you can follow the instructions for installing Snakemake and then install scipy with `pip install scipy` and numpy with `pip install numpy`.

4.1.3 Editing the Yaml File

The last step that is needed before pipeline execution is to edit the `ancestry_prediction_scRNAseq.yaml` file per your system and files. We suggest copying this file to a new filename and editing the new one.

Required User Input

Required

The first section of the `ancestry_prediction_scRNAseq.yaml` file will require user input:

```
################################################################################
##### The following arguments are for indicating file locations on your system #####
################################################################################
refs:
  genome: hg38 ## hg38 or hg19; genome the sequencing data have been aligned to
  hg19_fasta: /path/to/hg19/reference/genome.fa ## Path to the reference hg19 fasta to␣
↪be used for remapping for freebayes demultiplexing steps. Ideally this would be the␣
↪same reference used for original mapping but any reference on the same genome with the␣
↪same 'chr' encoding will do
  hg38_fasta: /path/to/hg38/reference/genome.fa ## ONLY NEEDED IF DATA ORIGINALLY MAPPED␣
↪TO HG38; Path to the reference hg38 fasta to be used for remapping for freebayes␣
↪demultiplexing steps. Ideally this would be the same reference used for original␣
↪mapping but any reference on the same genome with the same 'chr' encoding will do

inputs:
  metadata_file: /path/to/samples_meta.tsv ## Sample metadata file that has two columns:
↪'Pool' and 'Individual. The Pool should be the exact names of the parent folders for␣
↪the scRNAseq output
  singularity_image: /path/to/singularity/image.sif ### The complete path to the␣
↪singularity image that has all the softwares
```

(continues on next page)

(continued from previous page)

```
  bind_path: /path ## List of paths to bind to Singularity. You can specify multiple␣
↪directories by adding a "," between them. Eg. ${DIRECTORY1},${DIRECTORY2}. Singularity␣
↪will bind the directory that you are running from + subfolders but will not be able to␣
↪find anything above unless it is in this argument
  scRNAseq_dir: /path/to/scRNAseq/parent/directory ### the parent directory that has␣
↪directories for each pool and the scRNA-seq output below it
  barcode_annotation_dir: /path/to/barcodes/annotation/directory ### The directory that␣
↪contains each of the barcode files with per-barcode annotation. The pool name needs to␣
↪be within the file name. these should be filtered to remove doublets and contain only␣
↪cells assigned to an individual
  common_snps: None ### Leave as None for first run of the pipeline. This will be the␣
↪file of SNPs common across all sites and samples. This will be generated by sending␣
↪your snp list files to Drew Neavin and the garvan institute (d.neavin@garvan.org.au)␣
↪to create a common list of snps.
  barcode_tag: "CB"

outputs:
  outdir: /path/to/parent/out/dir
```

Please update the contents to reflect your data and your system. Here is a more detailed explanation of each entry:

`refs`

- `genome` - This is the genome that your single cell data have been aligned to. This should be either 'hg38' or 'hg19'.
- `hg19_fasta` - Path to an hg19 (or GRCh37) fasta file that has the same chr encoding (*i.e.* chr1 or 1) as your aligned single cell data.
- `hg38_fasta` - ONLY NEEDED IF DATA ORIGINALLY MAPPED TO HG38!!! If your data was aligned to hg19 (GRCh37), you can leave this field unedited. Otherwise, provide the path to an hg38 (GRCh38) fasta file.

`inputs`

- `metadata_file` - Path to tab-separated file that has two columns and a header. The first column is the Pool ID. This should be the same Pool IDs used for the directories of your single cell results. The second column should have the individuals in each pool that you want processed. See example in *Data Preparation*.
- `singularity_image` - Path to singularity image that you downloaded in *Installation*.
- `bind_path` - Path(s) to be bound for the singularity image. Singularity by default only binds the directories and files only below where you execute the command from. Therefore, it won't be able to find any files that are elsewhere on your system. Bind as many directories as you need by separating with a comma to so all the files you need can be found.
- `scRNAseq_dir` - The directory that contains directories for each single cell pool below it. The pool names should match those in your metadata_file. The pipeline is built to search for bam files downstream of each pool folder. You may run in to issues if you have multiple bam files.
- `barcode_annotation_dir` - A directory that contains files for each of the annotated barcode files prepared as described in the *Data Preparation*.
- `common_snps` - As part of this project, the SNPs called from multiple sites will be combined to identfiy the common SNPs across all sites. Leave this as 'None' for the first pipeline run. After common SNPs are generated, provide the path to that file.
- `barcode_tag` - The tag used to indicate the barcode in your bam file. For 10x, this will be "CB" but may be different for other technologies. Update as needed.

`outputs`

- `outdir` - The directory where you would like all results to be output.

Reference-Based Ancestry Predictions

Optional

We also provide the functionality to predict individual ancestries from reference SNP genotypes from microarrays or whole exome or genome sequencing data. As part of this, the pipeline will provide comparisons of the predictions between the reference and single cell data.

To implement reference-based ancestry predictions, you will need to provide information in the second section of the `ancestry_prediction_scRNAseq.yaml` file:

```
##########################################################################################
↪###################################
##### The following arguments are if you have reference SNPs for these individuals for␣
↪method accuracy testing purposes #####
##########################################################################################
↪###################################
snp:
  ref_snp_predict: False ## Set to true or false depending on if have reference SNP␣
↪genotype data to be predicted for pipeline accuracy testing purposes
  vcf: /path/to/unimputed/reference/vcf.vcf ## Reference SNP genotype vcf that is␣
↪UNIMPUTED
```

`ref_snp_predict` should be changed to True if you would like to predict ancestries based on reference SNP data

`vcf` should be the complete path for a vcf containing the SNP genotypes for each individual you would like to predict ancestry for.

Additional User Inputs

Each rule has different memory and thread allocations that can be altered and set by the user. You may want to update this if your jobs keep failing due to limited memory allocation. The important

This is an example fo the top few lines of this section:

```
##########################################################################################
↪###############################################
##### The following arguments are common parameters such as memory and threads that may␣
↪need to be changed depending on the dataset #####
##########################################################################################
↪###############################################
freebayes_ancestry:
  ### Following parameters are for bam subsetting by individual - will only be used if␣
↪multi-individual sample multiplexing was used
  subset_bam_memory: 4
  subset_bam_threads: 8

  ### Following parameters are for indexing the individual subset bam
  index_memory: 4
  index_threads: 2
```

4.1.4 Support

If you have any questions, suggestions or issues with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue or email Drew Neavin (d.neavin @ garvan.org.au)

4.1.5 Next

Now you should be ready to run the ancestry prediction pipline - proceed to *Ancestry Prediction Pipeline Execution*.

4.2 Ancestry Prediction Pipeline Execution

If you have any questions or issues, feel free to open an issue or directly email Drew Neavin (d.neavin @ garvan.org.au)

This section will provide commands to run the pipeline and expected results.

A detailed explanation of each step in the pipeline is provided in the *Manually Execute Pipeline* documents.

4.2.1 Pipeline Execution Phases

Phase 1

The first phase will identify the SNPs from each individual in each pool in your dataset. Then, we ask that you send that SNP list to Drew Neavin (d.neavin @ garvan.org.au) and she will identify the variants in common across all sites. She will send the list of SNPs common to all sites to each user.

Phase 2

The second phase will use the list of SNPs common to all sites to predict ancestries for each individual in the dataset.

The pipeline has been set up to run to the 'pause' point and then to execute the remainder of the rules once the SNPs common to all sites has been provided.

Optional Reference Ancestry Prediction

In addition, we have provided additional functionality to predict SNP-based ancestry from reference SNP genotypes (from microarrays or whole exome or genome sequencing). This also includes an automatic comparison between the reference and single cell-predicted ancestries for accuracy testing purposes. This is not dependent on the common SNP list across sites so can be executed at any time. See the *Preparing to Execute the Pipeline* documentation for more information on setting up your `ancestry_prediction_scRNAseq.yaml` file for this functionality.

4.2.2 Preparation

Before running Snakemake, let's define some variables that will make it easier and cleaner to run.

Let's define two variables, `$SNAKEFILE_ANC` (the location of the `Snakefile`) and `$CONFIG_ANC` which is (the location of the edited `ancestry_prediction_scRNAseq.yaml`). These files were copied from the singularity image when you ran the setup command:

```
.
├── ancestry_prediction_scRNAseq.yaml
├── includes
│   ├── reference_ancestry_predictions.smk
│   └── souporcell_ancestry.smk
├── mods
│   └── prepareArguments.py
├── Snakefile
└── snakemake.yaml
```

Change the path based on where the files is on your system:

```
SNAKEFILE_ANC=/path/to/ancestry_prediction_scRNAseq/Snakefile
CONFIG_ANC=/path/to/edited/ancestry_prediction_scRNAseq.yaml
```

Finally, let's define a location that we would like to keep the cluster log outputs and make the directory. Change the path based on where you want log files to be written.

```
LOG=/path/to/cluster/log_dir
mkdir -p $LOG
```

Important: If you log out of the cluster and log back in to run more jobs, you will have to redefine each of those variables. We recommend keeping the commands in a file that can easily be used to define each variable or putting them in a file that you can `source` before running Snakemake each time.

4.2.3 Running the Pipeline

Phase 1: SNP Calling

Now we have all the pieces that we need to run the pipeline. This Snakemake pipeline is built to run all the SNP genotype imputation pre-processing steps that are necessary. As discussed previously, the pipeline will be run in two phases. The first phase will identify the genetic variants in each sample. If you have reference SNP genotypes (*i.e.* microarray SNP genotype data or whole genome or exome data), some of those jobs will be run here too

1. First, let's do a "dry run" to identify what jobs will be run (remember to activate you snakemake environment before running: `conda activate ancestry_pred_snakemake`):

   ```
   snakemake \
       --snakefile $SNAKEFILE_ANC \
       --configfile $CONFIG_ANC \
       --dryrun \
       --cores 1 \
       --quiet
   ```

   The result should show you all the jobs that snakemake will run:

Without Reference SNP Genotypes

```
Job counts:
      count   jobs
      1       all
      1       common_snps_across_pools
      132     freebayes
      6       freebayes_common_snps
      6       freebayes_merge
      6       freebayes_update_vcf
      6       freebayes_vcf2plink
      6       index
      1       subset_bam
      165
```

With Reference SNP Genotypes :sync: key2

```
Job counts:
    count   jobs
    1       all
    1       common_snps_across_pools
    132     freebayes
    6       freebayes_common_snps
    6       freebayes_merge
    6       freebayes_update_vcf
    6       freebayes_vcf2plink
    6       index
    1       reference_common_snps
    1       reference_final_pruning
    1       reference_pca_1000g
    1       reference_pca_project
    1       reference_pca_projection_assign_original
    1       reference_prune_1000g
    1       reference_vcf2plink
    1       subset_bam
    172
```

Note

The number of rules to be run will depend on the number of samples and pools that you have. The number of `subset_bam` rules should reflect the number of pools you have. The number of all other rules should be the number of samples you have.

2. Next we can check how each of these jobs relates to one another:

```
snakemake \
    --snakefile $SNAKEFILE_ANC \
    --configfile $CONFIG_ANC \
    --dag | \
    dot -Tsvg \
        > dag1.svg
```

Without Reference SNP Genotypes

The resulting image will be saved to your current directory. In this case, we are using just one pool with 6 individuals for illustration purposes but this figure will change depending on the number of pools and individuals in your dataset. There's quite a lot in this figure so if you would like to see it you can view it here.

With Reference SNP Genotypes

The resulting image will be saved to your current directory. In this case, we are using just one pool with 6 individuals for illustration purposes but this figure will change depending on the number of pools and individuals in your dataset. There's quite a lot in this figure so if you would like to see it you can view it here.

3. Next, let's run those jobs:

Important

You will likely need to change the cluster command dependent on your job submission platform. This example is the job submission command for an SGE cluster. Some other submission examples for SLURM, LSF and SGE clusters are available in *Submission Examples* documentation.

```
nohup \
  snakemake \
    --snakefile $SNAKEFILE_ANC \
    --configfile $CONFIG_ANC \
    --rerun-incomplete \
    --jobs 20 \
    --use-singularity \
    --restart-times 2 \
    --keep-going \
    --cluster \
        "qsub -S /bin/bash \
        -q short.q \
        -r yes \
        -pe smp {threads} \
        -l tmp_requested={resources.disk_per_thread_gb}G \
        -l mem_requested={resources.mem_per_thread_gb}G \
        -e $LOG \
        -o $LOG \
        -j y \
        -V" \
  > $LOG/nohup_`date +%Y-%m-%d.%H:%M:%S`.log &
```

Expected Timing

~12-48 hours to run depending on the number of cells per individual and the coverage of SNPs

PAUSE

PAUSE

Send the resulting SNP file (`common_snps_across_pools.tsv`) which should be in your base output directory to Drew Neavin at d.neavin @ garvan.org.au so that SNPs common across all sites can be used for ancestry annotation. You will need to wait until you receive the file that contains common SNPs across each site.

Phase 2: Ancestry Prediction

Preparation

After you have received the list of SNP genotypes that were identified at each site from Drew, you will have to add the location of this file to your `ancestry_prediction_scRNAseq` in `common_snps:` (highlighted below):

```
##############################################################################################
##### The following arguments are for indicating file locations on your system #####
##############################################################################################
refs:
  genome: hg38 ## hg38 or hg19; genome the sequencing data have been aligned to
  hg19_fasta: /path/to/hg19/reference/genome.fa ## Path to the reference hg19 fasta to␣
↪be used for remapping for freebayes demultiplexing steps. Ideally this would be the␣
↪same reference used for original mapping but any reference on the same genome with the␣
↪same 'chr' encoding will do
  hg38_fasta: /path/to/hg38/reference/genome.fa ## ONLY NEEDED IF DATA ORIGINALLY MAPPED␣
↪TO HG38; Path to the reference hg38 fasta to be used for remapping for freebayes␣
↪demultiplexing steps. Ideally this would be the same reference used for original␣
↪mapping but any reference on the same genome with the same 'chr' encoding will do

inputs:
  metadata_file: /path/to/samples_meta.tsv ## Sample metadata file that has two columns:
↪'Pool' and 'N'. The Pool should be the exact names of the parent folders for the␣
↪scRNAseq output
  singularity_image: /path/to/singularity/image.sif ### The complete path to the␣
↪singularity image that has all the softwares
  bind_path: /path ## List of paths to bind to Singularity. You can specify multiple␣
↪directories by adding a "," between them. Eg. ${DIRECTORY1},${DIRECTORY2}. Singularity␣
↪will bind the directory that you are running from + subfolders but will not be able to␣
↪find anything above unless it is in this argument
  scRNAseq_dir: /path/to/scRNAseq/parent/directory ### the parent directory that has␣
↪directories for each pool and the scRNA-seq output below it
  barcode_annotation_dir: /path/to/barcodes/annotation/directory ### The directory that␣
↪contains each of the barcode files with per-barcode annotation. The pool name needs to␣
↪be within the file name. these should be filtered to remove doublets and contain only␣
↪cells assigned to an individual
  common_snps: /path/to/common_snps_across_sites.tsv ### Leave as None for first run of␣
↪the pipeline. This will be the file of SNPs common across all sites and samples. This␣
↪will be generated by sending your snp list files to Drew Neavin and the garvan␣
↪institute (d.neavin@garvan.org.au) to create a common list of snps.
  barcode_tag: "CB"
```

(continues on next page)

(continued from previous page)

```
outputs:
  outdir: /path/to/parent/out/dir
```

Execution

Now that we have provided the path to the SNP genotypes that will be used for ancestrys predictions, we can move on to execute the file steps of the pipeline:

1. Let's first do another dry run to see what steps will be run.

```
snakemake \
  --snakefile $SNAKEFILE_ANC \
  --configfile $CONFIG_ANC \
  --dryrun \
  --cores 1 \
  --reason
```

   - The result should show you all the jobs that snakemake will run:

Without Reference SNP Genotypes

```
Job counts:
    count   jobs
    1       all
    1       freebayes_combine_results
    6       freebayes_final_pruning
    1       freebayes_pca_1000g
    6       freebayes_pca_project
    6       freebayes_pca_projection_assign_original
    6       freebayes_prune_1000g
    1       subset_common_snps
    28
```

With Reference SNP Genotypes

```
Job counts:
  count jobs
  1             all
  1             freebayes_combine_results
  6             freebayes_final_pruning
  1             freebayes_pca_1000g
  6             freebayes_pca_project
  6             freebayes_pca_projection_assign_original
  6             freebayes_prune_1000g
  1             reference_freebayes_comparison
  1             subset_common_snps
  29
```

Note

The number of rules to be run will depend on the number of samples and pools that you have. The number for each rule should be the number of samples that you have. For this example we have oine pool that has 6 total samples.

2. Let's also take a look at how those new jobs fit in with the steps that we already ran:

```
snakemake \
    --snakefile $SNAKEFILE_ANC \
    --configfile $CONFIG_ANC \
    --dag | \
    dot -Tsvg \
        > dag2.svg
```

Without Reference SNP Genotypes

The resulting image will show jobs that are completed in dashed lines and those that still need to be run in solid lines. This will be saved to your current directory. The resulting saved image will show jobs that are completed in dashed lines and those that still need to be run in solid lines. In this case, we are using just one pool with 6 individuals for illustration purposes but this figure will change depending on the number of pools and individuals in your dataset. There's quite a lot in this figure so if you would like to see it you can view it here.

With Reference SNP Genotypes

The resulting image will show jobs that are completed in dashed lines and those that still need to be run in solid lines. This will be saved to your current directory. The resulting saved image will show jobs that are completed in dashed lines and those that still need to be run in solid lines. In this case, we are using just one pool with 6 individuals for illustration purposes but this figure will change depending on the number of pools and individuals in your dataset. There's quite a lot in this figure so if you would like to see it you can view it here.

3. Next, let's run those new jobs:

Note

Remember that you may need to change the cluster command dependent on your job submission platform. This example is the job submission command for an SGE cluster.

```
nohup \
  snakemake \
    --snakefile $SNAKEFILE_ANC \
    --configfile $CONFIG_ANC \
    --rerun-incomplete \
    --jobs 20 \
    --use-singularity \
    --restart-times 2 \
    --keep-going \
    --cluster \
        "qsub -S /bin/bash \
        -q short.q \
        -r yes \
        -pe smp {threads} \
        -l tmp_requested={resources.disk_per_thread_gb}G \
```

(continues on next page)

(continued from previous page)

```
        -l mem_requested={resources.mem_per_thread_gb}G \
        -e $LOG \
        -o $LOG \
        -j y \
        -V" \
  > $LOG/nohup_`date +%Y-%m-%d.%H:%M:%S`.log &
```

Results

Now you have run the complete pipeline and all of your results should be in the output directory that you indicated. An explanation of the results are in the *Results!* documentation.

4.3 Results!

After running those jobs, you should be done!

4.3.1 Overview of the Results

The following results should be in your output direcotry. The first tab provides a truncated results tree to help with visualization but you can see the complete tree from one pool with 6 samples in the second tab (Complete Tree).

Truncated Tree

Complete Tree

```
.
├── ancestry_assignments.tsv
├── Ancestry_PCAs.png
├── common_snps_across_pools.tsv
├── file_directories.txt
├── Pool1
│   ├── bams
│   │   ├── 1.bam
│   │   ├── 1.bam.bai
│   │   ├── 2.bam
│   │   ├── 2.bam.bai
│   │   ├── 3.bam
│   │   ├── 3.bam.bai
│   │   ├── 4.bam
│   │   ├── 4.bam.bai
│   │   ├── 5.bam
│   │   ├── 5.bam.bai
│   │   ├── 6.bam
│   │   ├── 6.bam.bai
│   │   └── subset_bam.done
│   ├── individual_1
│   │   ├── common_snps
│   │   │   ├── final_subset_pruned_data.bed
│   │   │   ├── final_subset_pruned_data.bim
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── final_subset_pruned_data.fam
│   │   │   ├── final_subset_pruned_data.log
│   │   │   ├── final_subset_pruned_data.pgen
│   │   │   ├── final_subset_pruned_data.psam
│   │   │   ├── final_subset_pruned_data.pvar
│   │   │   ├── snps_1000g.tsv
│   │   │   ├── SNPs2keep.txt
│   │   │   ├── snps_data.tsv
│   │   │   ├── subset_1000g.log
│   │   │   ├── subset_1000g.pgen
│   │   │   ├── subset_1000g.psam
│   │   │   ├── subset_1000g.pvar
│   │   │   ├── subset_data.log
│   │   │   ├── subset_data.pgen
│   │   │   ├── subset_data.prune.out
│   │   │   ├── subset_data.psam
│   │   │   ├── subset_data.pvar
│   │   │   ├── subset_pruned_1000g.bed
│   │   │   ├── subset_pruned_1000g.bim
│   │   │   ├── subset_pruned_1000g.fam
│   │   │   ├── subset_pruned_1000g.log
│   │   │   ├── subset_pruned_1000g.pgen
│   │   │   ├── subset_pruned_1000g.popu
│   │   │   ├── subset_pruned_1000g.prune.in
│   │   │   ├── subset_pruned_1000g.prune.out
│   │   │   ├── subset_pruned_1000g.psam
│   │   │   ├── subset_pruned_1000g.pvar
│   │   │   ├── subset_pruned_data_1000g_key.txt
│   │   │   ├── subset_pruned_data.log
│   │   │   ├── subset_pruned_data_original.pvar
│   │   │   ├── subset_pruned_data.pgen
│   │   │   ├── subset_pruned_data.psam
│   │   │   ├── subset_pruned_data.pvar
│   │   │   └── subset_pruned_data_temp.pvar
│   │   ├── freebayes_hg19.vcf
│   │   ├── freebayes_hg19.vcf.unmap
│   │   ├── freebayes.log
│   │   ├── freebayes.pgen
│   │   ├── freebayes.psam
│   │   ├── freebayes.pvar
│   │   ├── freebayes.pvar_original
│   │   ├── freebayes_tmp.pvar
│   │   ├── freebayes.vcf
│   │   ├── pca_projection
│   │   │   ├── final_subset_pruned_data_pcs.log
│   │   │   ├── final_subset_pruned_data_pcs.sscore
│   │   │   ├── subset_pruned_1000g_pcs.acount
│   │   │   ├── subset_pruned_1000g_pcs.eigenval
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   │   │   ├── subset_pruned_1000g_pcs.log
│   │   │   ├── subset_pruned_1000g_pcs_projected.log
```

(continues on next page)

(continued from previous page)

```
│   │   │   └── subset_pruned_1000g_pcs_projected.sscore
│   │   └── pca_sex_checks_original
│   │       ├── ancestry_assignments.tsv
│   │       ├── Ancestry_PCAs.png
│   │       └── variables.tsv
│   ├── individual_2
│   │   ...
│   ├── individual_3
│   │   ...
│   ├── individual_4
│   │   ...
│   ├── individual_5
│   │   ├── common_snps
│   │   ...
│   └── individual_6
│   └── ...
└── snps_1000g_common_across_sites.tsv
```

```
.
├── ancestry_assignments.tsv
├── Ancestry_PCAs.png
├── common_snps_across_pools.tsv
├── file_directories.txt
├── Pool1
│   ├── bams
│   │   ├── 1.bam
│   │   ├── 1.bam.bai
│   │   ├── 2.bam
│   │   ├── 2.bam.bai
│   │   ├── 3.bam
│   │   ├── 3.bam.bai
│   │   ├── 4.bam
│   │   ├── 4.bam.bai
│   │   ├── 5.bam
│   │   ├── 5.bam.bai
│   │   ├── 6.bam
│   │   ├── 6.bam.bai
│   │   └── subset_bam.done
│   ├── individual_1
│   │   ├── common_snps
│   │   │   ├── final_subset_pruned_data.bed
│   │   │   ├── final_subset_pruned_data.bim
│   │   │   ├── final_subset_pruned_data.fam
│   │   │   ├── final_subset_pruned_data.log
│   │   │   ├── final_subset_pruned_data.pgen
│   │   │   ├── final_subset_pruned_data.psam
│   │   │   ├── final_subset_pruned_data.pvar
│   │   │   ├── snps_1000g.tsv
│   │   │   ├── SNPs2keep.txt
│   │   │   ├── snps_data.tsv
│   │   │   ├── subset_1000g.log
│   │   │   ├── subset_1000g.pgen
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── subset_1000g.psam
│   │   │   ├── subset_1000g.pvar
│   │   │   ├── subset_data.log
│   │   │   ├── subset_data.pgen
│   │   │   ├── subset_data.prune.out
│   │   │   ├── subset_data.psam
│   │   │   ├── subset_data.pvar
│   │   │   ├── subset_pruned_1000g.bed
│   │   │   ├── subset_pruned_1000g.bim
│   │   │   ├── subset_pruned_1000g.fam
│   │   │   ├── subset_pruned_1000g.log
│   │   │   ├── subset_pruned_1000g.pgen
│   │   │   ├── subset_pruned_1000g.popu
│   │   │   ├── subset_pruned_1000g.prune.in
│   │   │   ├── subset_pruned_1000g.prune.out
│   │   │   ├── subset_pruned_1000g.psam
│   │   │   ├── subset_pruned_1000g.pvar
│   │   │   ├── subset_pruned_data_1000g_key.txt
│   │   │   ├── subset_pruned_data.log
│   │   │   ├── subset_pruned_data_original.pvar
│   │   │   ├── subset_pruned_data.pgen
│   │   │   ├── subset_pruned_data.psam
│   │   │   ├── subset_pruned_data.pvar
│   │   │   └── subset_pruned_data_temp.pvar
│   │   ├── freebayes_hg19.vcf
│   │   ├── freebayes_hg19.vcf.unmap
│   │   ├── freebayes.log
│   │   ├── freebayes.pgen
│   │   ├── freebayes.psam
│   │   ├── freebayes.pvar
│   │   ├── freebayes.pvar_original
│   │   ├── freebayes_tmp.pvar
│   │   ├── freebayes.vcf
│   │   ├── pca_projection
│   │   │   ├── final_subset_pruned_data_pcs.log
│   │   │   ├── final_subset_pruned_data_pcs.sscore
│   │   │   ├── subset_pruned_1000g_pcs.acount
│   │   │   ├── subset_pruned_1000g_pcs.eigenval
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   │   │   ├── subset_pruned_1000g_pcs.log
│   │   │   ├── subset_pruned_1000g_pcs_projected.log
│   │   │   └── subset_pruned_1000g_pcs_projected.sscore
│   │   └── pca_sex_checks_original
│   │       ├── ancestry_assignments.tsv
│   │       ├── Ancestry_PCAs.png
│   │       └── variables.tsv
│   ├── individual_2
│   │   ├── common_snps
│   │   │   ├── final_subset_pruned_data.bed
│   │   │   ├── final_subset_pruned_data.bim
│   │   │   ├── final_subset_pruned_data.fam
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── final_subset_pruned_data.log
│   │   │   ├── final_subset_pruned_data.pgen
│   │   │   ├── final_subset_pruned_data.psam
│   │   │   ├── final_subset_pruned_data.pvar
│   │   │   ├── snps_1000g.tsv
│   │   │   ├── SNPs2keep.txt
│   │   │   ├── snps_data.tsv
│   │   │   ├── subset_1000g.log
│   │   │   ├── subset_1000g.pgen
│   │   │   ├── subset_1000g.psam
│   │   │   ├── subset_1000g.pvar
│   │   │   ├── subset_data.log
│   │   │   ├── subset_data.pgen
│   │   │   ├── subset_data.prune.out
│   │   │   ├── subset_data.psam
│   │   │   ├── subset_data.pvar
│   │   │   ├── subset_pruned_1000g.bed
│   │   │   ├── subset_pruned_1000g.bim
│   │   │   ├── subset_pruned_1000g.fam
│   │   │   ├── subset_pruned_1000g.log
│   │   │   ├── subset_pruned_1000g.pgen
│   │   │   ├── subset_pruned_1000g.popu
│   │   │   ├── subset_pruned_1000g.prune.in
│   │   │   ├── subset_pruned_1000g.prune.out
│   │   │   ├── subset_pruned_1000g.psam
│   │   │   ├── subset_pruned_1000g.pvar
│   │   │   ├── subset_pruned_data_1000g_key.txt
│   │   │   ├── subset_pruned_data.log
│   │   │   ├── subset_pruned_data_original.pvar
│   │   │   ├── subset_pruned_data.pgen
│   │   │   ├── subset_pruned_data.psam
│   │   │   ├── subset_pruned_data.pvar
│   │   │   └── subset_pruned_data_temp.pvar
│   │   ├── freebayes_hg19.vcf
│   │   ├── freebayes_hg19.vcf.unmap
│   │   ├── freebayes.log
│   │   ├── freebayes.pgen
│   │   ├── freebayes.psam
│   │   ├── freebayes.pvar
│   │   ├── freebayes.pvar_original
│   │   ├── freebayes_tmp.pvar
│   │   ├── freebayes.vcf
│   │   ├── pca_projection
│   │   │   ├── final_subset_pruned_data_pcs.log
│   │   │   ├── final_subset_pruned_data_pcs.sscore
│   │   │   ├── subset_pruned_1000g_pcs.acount
│   │   │   ├── subset_pruned_1000g_pcs.eigenval
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   │   │   ├── subset_pruned_1000g_pcs.log
│   │   │   ├── subset_pruned_1000g_pcs_projected.log
│   │   │   └── subset_pruned_1000g_pcs_projected.sscore
```

(continues on next page)

(continued from previous page)

```
│   │   └── pca_sex_checks_original
│   │       ├── ancestry_assignments.tsv
│   │       ├── Ancestry_PCAs.png
│   │       └── variables.tsv
│   ├── individual_3
│   │   ├── common_snps
│   │   │   ├── final_subset_pruned_data.bed
│   │   │   ├── final_subset_pruned_data.bim
│   │   │   ├── final_subset_pruned_data.fam
│   │   │   ├── final_subset_pruned_data.log
│   │   │   ├── final_subset_pruned_data.pgen
│   │   │   ├── final_subset_pruned_data.psam
│   │   │   ├── final_subset_pruned_data.pvar
│   │   │   ├── snps_1000g.tsv
│   │   │   ├── SNPs2keep.txt
│   │   │   ├── snps_data.tsv
│   │   │   ├── subset_1000g.log
│   │   │   ├── subset_1000g.pgen
│   │   │   ├── subset_1000g.psam
│   │   │   ├── subset_1000g.pvar
│   │   │   ├── subset_data.log
│   │   │   ├── subset_data.pgen
│   │   │   ├── subset_data.prune.out
│   │   │   ├── subset_data.psam
│   │   │   ├── subset_data.pvar
│   │   │   ├── subset_pruned_1000g.bed
│   │   │   ├── subset_pruned_1000g.bim
│   │   │   ├── subset_pruned_1000g.fam
│   │   │   ├── subset_pruned_1000g.log
│   │   │   ├── subset_pruned_1000g.pgen
│   │   │   ├── subset_pruned_1000g.popu
│   │   │   ├── subset_pruned_1000g.prune.in
│   │   │   ├── subset_pruned_1000g.prune.out
│   │   │   ├── subset_pruned_1000g.psam
│   │   │   ├── subset_pruned_1000g.pvar
│   │   │   ├── subset_pruned_data_1000g_key.txt
│   │   │   ├── subset_pruned_data.log
│   │   │   ├── subset_pruned_data_original.pvar
│   │   │   ├── subset_pruned_data.pgen
│   │   │   ├── subset_pruned_data.psam
│   │   │   ├── subset_pruned_data.pvar
│   │   │   └── subset_pruned_data_temp.pvar
│   │   ├── freebayes_hg19.vcf
│   │   ├── freebayes_hg19.vcf.unmap
│   │   ├── freebayes.log
│   │   ├── freebayes.pgen
│   │   ├── freebayes.psam
│   │   ├── freebayes.pvar
│   │   ├── freebayes.pvar_original
│   │   ├── freebayes_tmp.pvar
│   │   ├── freebayes.vcf
│   │   ├── pca_projection
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── final_subset_pruned_data_pcs.log
│   │   │   ├── final_subset_pruned_data_pcs.sscore
│   │   │   ├── subset_pruned_1000g_pcs.acount
│   │   │   ├── subset_pruned_1000g_pcs.eigenval
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   │   │   ├── subset_pruned_1000g_pcs.log
│   │   │   ├── subset_pruned_1000g_pcs_projected.log
│   │   │   └── subset_pruned_1000g_pcs_projected.sscore
│   │   └── pca_sex_checks_original
│   │       ├── ancestry_assignments.tsv
│   │       ├── Ancestry_PCAs.png
│   │       └── variables.tsv
│   ├── individual_4
│   │   ├── common_snps
│   │   │   ├── final_subset_pruned_data.bed
│   │   │   ├── final_subset_pruned_data.bim
│   │   │   ├── final_subset_pruned_data.fam
│   │   │   ├── final_subset_pruned_data.log
│   │   │   ├── final_subset_pruned_data.pgen
│   │   │   ├── final_subset_pruned_data.psam
│   │   │   ├── final_subset_pruned_data.pvar
│   │   │   ├── snps_1000g.tsv
│   │   │   ├── SNPs2keep.txt
│   │   │   ├── snps_data.tsv
│   │   │   ├── subset_1000g.log
│   │   │   ├── subset_1000g.pgen
│   │   │   ├── subset_1000g.psam
│   │   │   ├── subset_1000g.pvar
│   │   │   ├── subset_data.log
│   │   │   ├── subset_data.pgen
│   │   │   ├── subset_data.prune.out
│   │   │   ├── subset_data.psam
│   │   │   ├── subset_data.pvar
│   │   │   ├── subset_pruned_1000g.bed
│   │   │   ├── subset_pruned_1000g.bim
│   │   │   ├── subset_pruned_1000g.fam
│   │   │   ├── subset_pruned_1000g.log
│   │   │   ├── subset_pruned_1000g.pgen
│   │   │   ├── subset_pruned_1000g.popu
│   │   │   ├── subset_pruned_1000g.prune.in
│   │   │   ├── subset_pruned_1000g.prune.out
│   │   │   ├── subset_pruned_1000g.psam
│   │   │   ├── subset_pruned_1000g.pvar
│   │   │   ├── subset_pruned_data_1000g_key.txt
│   │   │   ├── subset_pruned_data.log
│   │   │   ├── subset_pruned_data_original.pvar
│   │   │   ├── subset_pruned_data.pgen
│   │   │   ├── subset_pruned_data.psam
│   │   │   ├── subset_pruned_data.pvar
│   │   │   └── subset_pruned_data_temp.pvar
│   │   ├── freebayes_hg19.vcf
```

(continues on next page)

(continued from previous page)

```
│   │   ├── freebayes_hg19.vcf.unmap
│   │   ├── freebayes.log
│   │   ├── freebayes.pgen
│   │   ├── freebayes.psam
│   │   ├── freebayes.pvar
│   │   ├── freebayes.pvar_original
│   │   ├── freebayes_tmp.pvar
│   │   ├── freebayes.vcf
│   │   ├── pca_projection
│   │   │   ├── final_subset_pruned_data_pcs.log
│   │   │   ├── final_subset_pruned_data_pcs.sscore
│   │   │   ├── subset_pruned_1000g_pcs.acount
│   │   │   ├── subset_pruned_1000g_pcs.eigenval
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   │   │   ├── subset_pruned_1000g_pcs.log
│   │   │   ├── subset_pruned_1000g_pcs_projected.log
│   │   │   └── subset_pruned_1000g_pcs_projected.sscore
│   │   └── pca_sex_checks_original
│   │       ├── ancestry_assignments.tsv
│   │       ├── Ancestry_PCAs.png
│   │       └── variables.tsv
│   ├── individual_5
│   │   ├── common_snps
│   │   │   ├── final_subset_pruned_data.bed
│   │   │   ├── final_subset_pruned_data.bim
│   │   │   ├── final_subset_pruned_data.fam
│   │   │   ├── final_subset_pruned_data.log
│   │   │   ├── final_subset_pruned_data.pgen
│   │   │   ├── final_subset_pruned_data.psam
│   │   │   ├── final_subset_pruned_data.pvar
│   │   │   ├── snps_1000g.tsv
│   │   │   ├── SNPs2keep.txt
│   │   │   ├── snps_data.tsv
│   │   │   ├── subset_1000g.log
│   │   │   ├── subset_1000g.pgen
│   │   │   ├── subset_1000g.psam
│   │   │   ├── subset_1000g.pvar
│   │   │   ├── subset_data.log
│   │   │   ├── subset_data.pgen
│   │   │   ├── subset_data.prune.out
│   │   │   ├── subset_data.psam
│   │   │   ├── subset_data.pvar
│   │   │   ├── subset_pruned_1000g.bed
│   │   │   ├── subset_pruned_1000g.bim
│   │   │   ├── subset_pruned_1000g.fam
│   │   │   ├── subset_pruned_1000g.log
│   │   │   ├── subset_pruned_1000g.pgen
│   │   │   ├── subset_pruned_1000g.popu
│   │   │   ├── subset_pruned_1000g.prune.in
│   │   │   ├── subset_pruned_1000g.prune.out
│   │   │   ├── subset_pruned_1000g.psam
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── subset_pruned_1000g.pvar
│   │   │   ├── subset_pruned_data_1000g_key.txt
│   │   │   ├── subset_pruned_data.log
│   │   │   ├── subset_pruned_data_original.pvar
│   │   │   ├── subset_pruned_data.pgen
│   │   │   ├── subset_pruned_data.psam
│   │   │   ├── subset_pruned_data.pvar
│   │   │   └── subset_pruned_data_temp.pvar
│   │   ├── freebayes_hg19.vcf
│   │   ├── freebayes_hg19.vcf.unmap
│   │   ├── freebayes.log
│   │   ├── freebayes.pgen
│   │   ├── freebayes.psam
│   │   ├── freebayes.pvar
│   │   ├── freebayes.pvar_original
│   │   ├── freebayes_tmp.pvar
│   │   ├── freebayes.vcf
│   │   ├── pca_projection
│   │   │   ├── final_subset_pruned_data_pcs.log
│   │   │   ├── final_subset_pruned_data_pcs.sscore
│   │   │   ├── subset_pruned_1000g_pcs.acount
│   │   │   ├── subset_pruned_1000g_pcs.eigenval
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec
│   │   │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   │   │   ├── subset_pruned_1000g_pcs.log
│   │   │   ├── subset_pruned_1000g_pcs_projected.log
│   │   │   └── subset_pruned_1000g_pcs_projected.sscore
│   │   └── pca_sex_checks_original
│   │       ├── ancestry_assignments.tsv
│   │       ├── Ancestry_PCAs.png
│   │       └── variables.tsv
│   └── individual_6
│       ├── common_snps
│       │   ├── final_subset_pruned_data.bed
│       │   ├── final_subset_pruned_data.bim
│       │   ├── final_subset_pruned_data.fam
│       │   ├── final_subset_pruned_data.log
│       │   ├── final_subset_pruned_data.pgen
│       │   ├── final_subset_pruned_data.psam
│       │   ├── final_subset_pruned_data.pvar
│       │   ├── snps_1000g.tsv
│       │   ├── SNPs2keep.txt
│       │   ├── snps_data.tsv
│       │   ├── subset_1000g.log
│       │   ├── subset_1000g.pgen
│       │   ├── subset_1000g.psam
│       │   ├── subset_1000g.pvar
│       │   ├── subset_data.log
│       │   ├── subset_data.pgen
│       │   ├── subset_data.prune.out
│       │   ├── subset_data.psam
│       │   ├── subset_data.pvar
```

(continues on next page)

(continued from previous page)

```
│       │   ├── subset_pruned_1000g.bed
│       │   ├── subset_pruned_1000g.bim
│       │   ├── subset_pruned_1000g.fam
│       │   ├── subset_pruned_1000g.log
│       │   ├── subset_pruned_1000g.pgen
│       │   ├── subset_pruned_1000g.popu
│       │   ├── subset_pruned_1000g.prune.in
│       │   ├── subset_pruned_1000g.prune.out
│       │   ├── subset_pruned_1000g.psam
│       │   ├── subset_pruned_1000g.pvar
│       │   ├── subset_pruned_data_1000g_key.txt
│       │   ├── subset_pruned_data.log
│       │   ├── subset_pruned_data_original.pvar
│       │   ├── subset_pruned_data.pgen
│       │   ├── subset_pruned_data.psam
│       │   ├── subset_pruned_data.pvar
│       │   └── subset_pruned_data_temp.pvar
│       ├── freebayes_hg19.vcf
│       ├── freebayes_hg19.vcf.unmap
│       ├── freebayes.log
│       ├── freebayes.pgen
│       ├── freebayes.psam
│       ├── freebayes.pvar
│       ├── freebayes.pvar_original
│       ├── freebayes_tmp.pvar
│       ├── freebayes.vcf
│       ├── pca_projection
│       │   ├── final_subset_pruned_data_pcs.log
│       │   ├── final_subset_pruned_data_pcs.sscore
│       │   ├── subset_pruned_1000g_pcs.acount
│       │   ├── subset_pruned_1000g_pcs.eigenval
│       │   ├── subset_pruned_1000g_pcs.eigenvec
│       │   ├── subset_pruned_1000g_pcs.eigenvec.allele
│       │   ├── subset_pruned_1000g_pcs.log
│       │   ├── subset_pruned_1000g_pcs_projected.log
│       │   └── subset_pruned_1000g_pcs_projected.sscore
│       └── pca_sex_checks_original
│           ├── ancestry_assignments.tsv
│           ├── Ancestry_PCAs.png
│           └── variables.tsv
└── snps_1000g_common_across_sites.tsv
```

4.3.2 Inforamtive Results

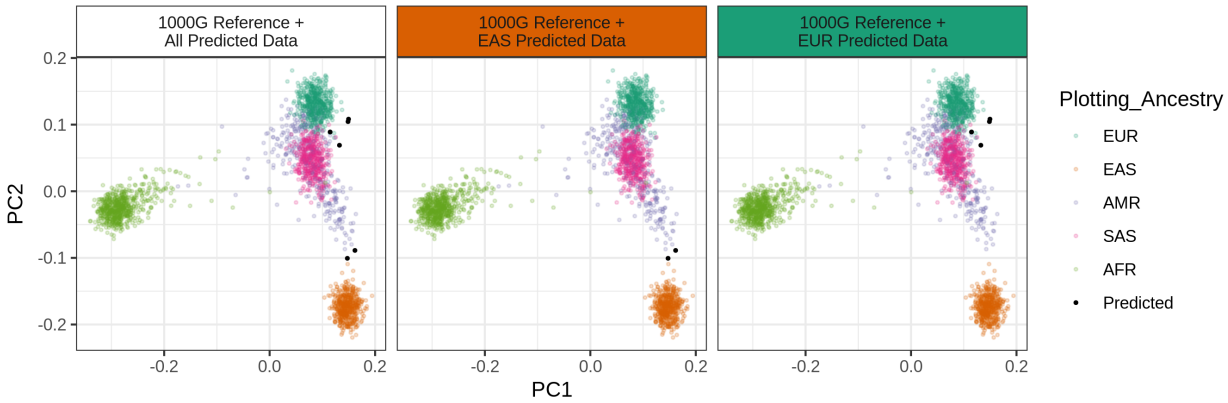
Your output directory will have summarized results in it (`ancestry_assignments.tsv` and `Ancestry_PCAs.png`). There are files of each individual in their separate direcotires in the `pca_sex_checks_original` directory (for example `/path/to/parent/out/dir/Pool1/individual_1/pca_sex_checks_original/Ancestry_PCAs.png`). Here's an example of each of these;

`ancestry_assignments.tsv`: This file contains the predicted ancestries for all the samples as well as all the principal component locations for each individual (this will be in your output directory `/path/to/parent/out/dir/ancestry_assignments.tsv`). The assignments are the second to last column:

FID	IID	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	AFR	AMR	EAS	EUR	SAS	Final_Assignment	Pool
0	1	0.162	-0.089	-0.041	-0.014	0.059	0.010	0.028	-0.015	0.011	0.019	0	0	1	0	0	EAS	RZ731_Pool8
0	2	0.115	0.089	0.037	-0.019	0.017	0.030	0.047	0.008	-0.017	-0.012	0	0	0	1	0	EUR	RZ731_Pool8
0	3	0.147	-0.101	-0.089	-0.001	0.116	0.014	0.006	0.037	0.027	-0.035	0	0	1	0	0	EAS	RZ731_Pool8
0	4	0.150	0.108	0.023	-0.062	0.029	0.024	-0.010	-0.008	-0.034	-0.013	0	0	0	1	0	EUR	RZ731_Pool8
0	5	0.132	0.069	-0.033	-0.029	0.045	0.017	0.020	-0.011	0.011	0.023	0	0.11	0	0.89	0	EUR	RZ731_Pool8
0	6	0.149	0.105	-0.022	-0.053	0.063	0.083	-0.014	-0.029	-0.020	0.008	0	0	0	1	0	EUR	RZ731_Pool8

`Ancestry_PCAs.png`: a separate figure generated for each individual in each pool. For example, a PCA plot for individual 1 in Pool 1: `/path/to/parent/out/dir/Pool1/individual_1/pca_sex_checks_original/Ancestry_PCAs.png`.

- This figure shows the 1000G individual locations in PC space compared to the individual. For example:



- There will be an `ancestry_assignments.tsv` file generated for each individual in each pool and one that has all the individuals joined together in the base output directory.
  - This file has the annotations and probabilities for each pool. For example:

FID	IID	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	AFR	AMR	EAS	EUR	SAS	combined_assignment	Final_Assignment
0	Pool1	0.137	-0.108	-0.025	-0.042	0.032	-0.042	0.001	-0.021	-0.086	-0.019	0	0	1	0	0	EAS	EAS

4.3.3 Additional Results with Reference-based Exectution

If you have reference SNP genotypes (*i.e.* microarray or whole exome or genome sequencing-called SNPs) and decided to estimate SNP-based ancestry, you will have additional results that compare the reference and single-cell ancestry predictions. These will be located in:

- `/path/to/parent/out/dir/reference`: This will contain the ancestry predictions for the reference-based ancestry predictions per individual.
- `/path/to/parent/out/dir/ref_sc_ancestry_prediction_comparison`: This will provide results on reference vs single cell based ancestry predictions. These are the main comparison files with the two most informative ones highlighted and additional information below:

```
snp_ancestry_predictions.tsv
reference_ancestry_numbers.png
predicted_ancestry_numbers_correct.png
predicted_ancestry_numbers_correct_identified.png
statistics_heatmap.png
predicted_numbers_statistics_heatmap_combined.png
assignments_probabilities_w_ref.png
```

The `predicted_numbers_statistics_heatmap_combined.png` figure shows the number of individuals classified to each ancestry with the single cell data and colored by if they correctly or incorrectly match the reference-annotated SNP genotype data and the heatmap below shows some statistical metrics for quantifying the single cell derived ancestry predictions:

The `assignments_probabilities_w_ref.png` figure show the probability of each sample to be classified to each of the different ancestries. This includes the reference-based predictions (microarray or whole exome or genome sequencing data) compared to the single cell based predictions

4.4 Submission Examples

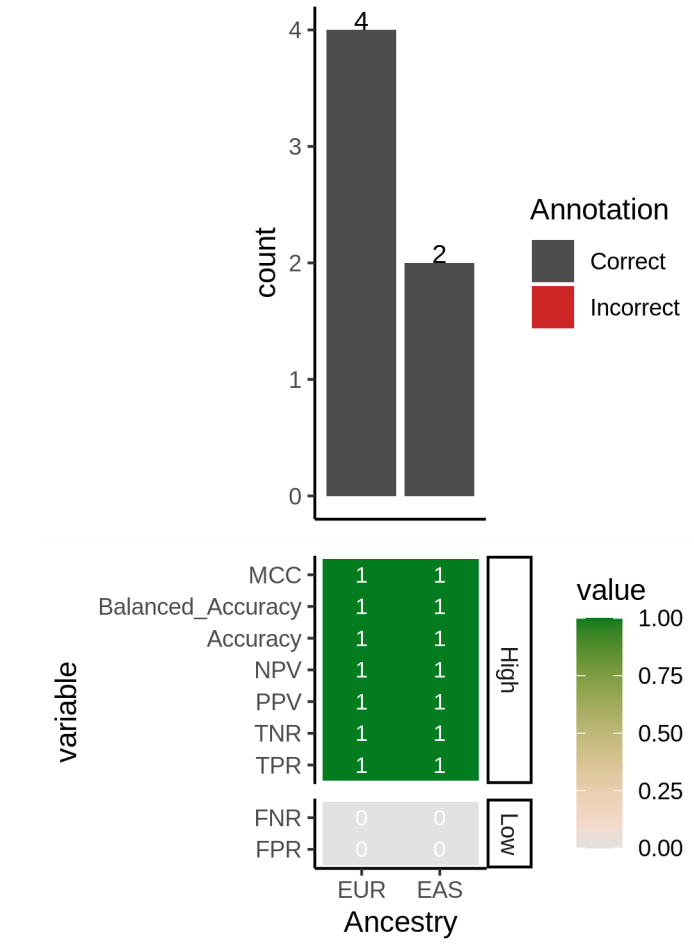
The submission command for snakemake is dependent on the cluster infrastructure so here are a few examples submission commands to help get you started. If you find a different submission works on your cluster and you think it could help others, feel free to send it to us so that we can add it here (either open an issue or email d.neavin @ garvan.org.au directly)

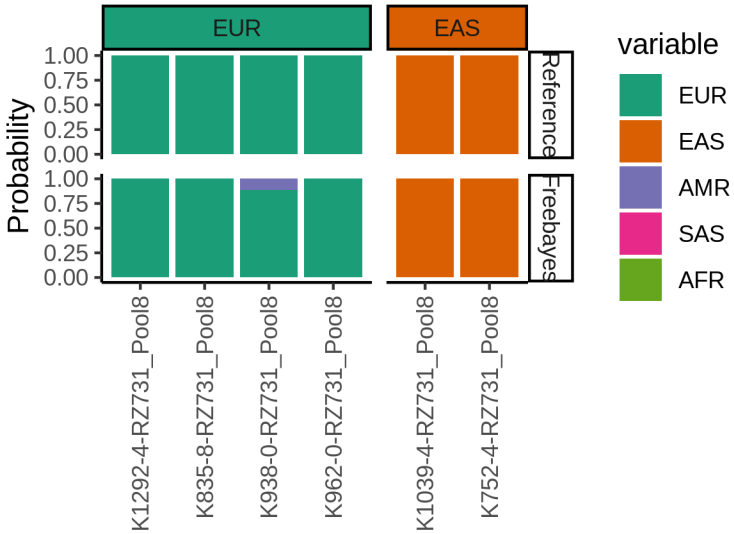
4.4.1 LSF Example

Here is an example of a submission for a LSF cluster. Of course, you may have to update it based on how your cluster has been set up.

```
nohup \
snakemake \
--snakefile $SCEQTL_PIPELINE_SNAKEFILE \
--configfile $SCEQTL_PIPELINE_CONFIG \
--rerun-incomplete \
--jobs 20 \
--use-singularity \
--restart-times 2 \
--keep-going \
--cluster \
```

(continues on next page)


count
4
2
Annotation
Correct
Incorrect
variable
MCC
Balanced_Accuracy
Accuracy
NPV
PPV
TNR
TPR
FNR
FPR
High
Low
value
1.00
0.75
0.50
0.25
0.00
EUR
EAS
Ancestry


EUR
EAS
Probability
1.00
0.75
0.50
0.25
0.00
Reference
Freebayes
variable
EUR
EAS
AMR
SAS
AFR
K1292-4-RZ731_Pool8
K835-8-RZ731_Pool8
K938-0-RZ731_Pool8
K962-0-RZ731_Pool8
K1039-4-RZ731_Pool8
K752-4-RZ731_Pool8

(continued from previous page)

```
  "bsub \
  -W 24:00 \
  -x \
  -M 10000 \
  -e $LOG \
  -o $LOG" \
> $LOG/nohup_`date +%Y-%m-%d.%H:%M:%S`.log &
```

4.4.2 SGE Example

This is an additional example for an SGE cluster.

```
nohup \
  snakemake \
    --snakefile $IMPUTATION_SNAKEFILE \
    --configfile $IMPUTATION_CONFIG \
    --rerun-incomplete \
    --jobs 20 \
    --use-singularity \
    --restart-times 2 \
    --keep-going \
    --cluster \
        "qsub -S /bin/bash \
        -q short.q \
        -r yes \
        -pe smp {threads} \
        -l tmp_requested={resources.disk_per_thread_gb}G \
        -l mem_requested={resources.mem_per_thread_gb}G \
        -e $LOG \
        -o $LOG \
        -j y \
        -V" \
  > $LOG/nohup_`date +%Y-%m-%d.%H:%M:%S`.log &
```

4.4.3 SLURM Examples

```
nohup \
  snakemake \
    --snakefile $IMPUTATION_SNAKEFILE \
    --configfile $IMPUTATION_CONFIG \
    --rerun-incomplete \
    --jobs 48 \
    --use-singularity \
    --restart-times 2 \
    --keep-going \
    --cluster \
       "sbatch \
       --qos debug \
       -N 1 \
```

(continues on next page)

(continued from previous page)

```
        --ntasks 1 \
        --cpus-per-task 48 \
        -o $LOG/%{rule}.out \
        --export ALL" \
      > $LOG/nohup_`date +%Y-%m-%d.%H:%M:%S`.log &
```

Another SLURM example where file latency causes issues with snakemakes ability to detect if a job is completed (note the `--latency-wait` parameter):

```
nohup
  snakemake \
    --snakefile $IMPUTATION_SNAKEFILE \
    --configfile $IMPUTATION_CONFIG \
    --rerun-incomplete \
    --jobs 1 \
    --use-singularity \
    --restart-times 2 \
    --keep-going \
    --latency-wait 30 \
    --cluster \
        "sbatch \
        --qos regular \
        -N {threads} \
        --mem={resources.mem_per_thread_gb}G \
        --tmp={resources.disk_per_thread_gb}G \
        -o $LOG/{rule}.out \
        --export ALL \
        --time=05:59:59" \
      > $LOG/nohup_`date +%Y-%m-%d.%H:%M:%S`.log &
```

4.5 Support

If you have any questions, suggestions or issues with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue or email Drew Neavin (d.neavin @ garvan.org.au)

CHAPTER

FIVE

EXECUTE STEPS MANUALLY

We have provided the commands to run each single cell capture manually if your data don't match the assumptions of the snakemake pipeline, you want to alter some parameters of you just want to understand the steps that are being executed at each step.

Importantly, all the softwares and files required to run these steps manually are still provided in the Singularity image so no additional instllation will be necessary.

To run your dataset(s) manually, procded to the *Manually Execute Pipeline* documentation.

5.1 Manually Execute Pipeline

This section shows each of the commands to run the ancestry prediction pipeline manually. You can run each step of the pipeline from the Singularity image downloaded in the *Installation* section which should help standardize execution of this pipeline across different computing systems.

Note

If you have multiple pools and/or multiple individuals in each pool, you will need to update the output directories to reflect this and parallelize accross each of these combinations (following bam subsetting if you multiplexed multiple individuals in a pool). The snakemake pipeline is designed to automatically parallelize these jobs and monitor the state of each job so we highly recommend leveraging the pipeline if you can do so with your data.

The general steps are:

1. *Divide Bam File by Individual* Optional
2. *Index Bam File(s)* Required
3. *Freebayes SNP Calling* Required
4. *Merge Freebayes Results into Single File* Required
5. *Lift hg38 to hg19* Optional
6. *Convert vcf to Plink* Required
7. *Identify Common SNPs Between 1000G Data and Your Data* Required
8. *Get the SNPs in Common Across all Pools* Required

PAUSE - send results to Drew Neavin (d.neavin @ garvan.org.au)

1. *Subset Data for Common SNPs* Required
2. *Prune the SNPs in LD* Required

3. *Filter 1000G SNPs for Pruned SNPs* Required
4. *Filter Freebayes SNPs for Pruned SNPs* Required
5. *Calculate PCs using 1000G* Required
6. *Project 1000G and Freebayes Data in PCs* Required
7. *Plot PC Results* Required

5.1.1 Setup

There are a few variables that will be used throughout the example commands which are best to define in a file that you can easily run at each step or source for execution of each step.

- `$BIND` - the path(s) on your system to bind to singularity when executing commands. By default, Singularity binds just the directory (and downstream dierectory and files) from your current working directory when executing the command. However, if you have some files elsewhere on your system, you can provide a parent directory to the singulairty command to indicate which additional directories to bind. Multiple directories can be included and separated by a comma (*i.e.* $DIR1,$DIR2)
- `$SIF` - the path to the singularity image downloaded in the *Installation* section
- `$OUTDIR` - the path to the output directory where all results will be written and saved

```
BIND = /bind/path
SIF = /path/to/singularity/image/ancestry_prediction_scRNAseq.sif
OUTDIR = /path/to/base/outdir
```

5.1.2 Steps

1. Divide Bam File

Optional

Expected Timing

~15-40 minutes when using 8 threads with 4G each

If you have multiple individuals per capture, you will need to do this step but if you only have one individual in your pool, you do not have to split the bam by individuals and you can proced directly to *2. Index Bam File(s)*

In preparation for this step, we set some additional parameters and create the required output directory. The parameters that we use for the command in this step are

```
BAM=/path/to/bam/file.bam ### Path to bam file
ANNO_BARCODES=/path/to/annotated/barcodes.tsv ### Path to annotated barcodes
TAG="CB"
N=8

mkdir -p $OUTDIR/bams
```

- The `$ANNO_BARCODES` is the annotated barcodes file described in *Data Preparation*
- The `$TAG` is the tag used in your bam file to indicate cell barcodes. In 10x captures, this is 'CB' but could be different for different technologies

To divide the bam file into a single file for each individual in the pool, simply execute:

```
singularity exec --bind $BIND $SIF sinto filterbarcodes -b $BAM -c $ANNO_BARCODES --
↪barcodetag $TAG --outdir $OUTDIR/bams --nproc $N
```

2. Index Bam File(s)

Required

Expected Timing

< 5 min

The bam file(s) need to be indexed before SNP calling with freebayes. Of course, if your bam is already indexed, you can skip to *Freebayes SNP Calling*

The `$BAM` will be either your original bam file (if did not subset by individual in previous step) or one of the bam files subset by each individual in the pool

```
singularity exec --bind $BIND $SIF samtools index $BAM
```

3. Freebayes SNP Calling

Required

Expected Timing

~12 and 36 hours with 8 to 16 threads with 4-16G each. The time for this step will vary greatly depending on the number of reads per capture captured and the number of cells per individual.

Freebayes will be used to call SNP genotypes from the bam file using known common SNP genotype locations based on 1000G data. In order to expedite this process, we suggest running each chromosome in parallel.

We have provided common SNP location bed files that can be used for calling SNPs with freebayes filtered by minor allele frequency for each chromosome. The files contain SNPs on either hg19/GRCh37 or hg38/GRCh38 and either have 'chr' encoding or not for each chromosome ('chr1' vs 1) and are located in the `/opt` directory in the `ancestry_prediction_scRNAseq.sif` image. You will be able to use these files from directly within the `ancestry_prediction_scRNAseq.sif` image. The table below explains the location that you should use depending on your data. The * indicates that there is a different file for each chromosome from 1 to 22.

Genome	Chr Encoding	vcf File
GRCh37	No 'chr'	/opt/GRCh37_1000G_MAF0.01_GeneFiltered_NoChr/GRCh37_1000G_MAF0.01_GeneFiltered_NoChr_*.bed
	'chr' encoding	/opt/GRCh37_1000G_MAF0.01_GeneFiltered_ChrEncoding/GRCh37_1000G_MAF0.01_GeneFiltered_ChrEncoding_chr*.bed
GRCh38	No 'chr'	/opt/GRCh38_1000G_MAF0.01_GeneFiltered_NoChr/GRCh38_1000G_MAF0.01_GeneFiltered_NoChr_*.bed
	'chr' encoding	/opt/GRCh38_1000G_MAF0.01_GeneFiltered_ChrEncoding/GRCh38_1000G_MAF0.01_GeneFiltered_ChrEncoding_chr*.bed

Define some variables to execute the freebayes SNP calling The `$TARGETS` is the the bed file containing the common SNP locations in the `ancestry_prediction_scRNAseq.sif` image. For example for chromosome 1:

```
N=8
TARGETS=/opt/GRCh37_1000G_MAF0.01_GeneFiltered_NoChr/GRCh37_1000G_MAF0.01_GeneFiltered_
↪NoChr_1.bed
FASTA=/path/to/reference/fasta.fa
```

Here's an example command to run freebayes to identify the SNP genotypes for the individual in the bam file for chromosome 1 but as we mentioned above, we suggest that you run each chromosome in parallel to expedite this step:

```
export TMPDIR=/tmp
singularity exec --bind $BIND $SIF freebayes -f $FASTA -iXu -C 2 -q 20 -n 3 -E 1 -m 30 --
↪min-coverage 6 --limit-coverage 100000 --targets $BED_DIR/GRCh38_1000G_MAF0.01_
↪GeneFiltered_NoChr_1.bed $BAM > $OUTDIR/freebayes_chr1.vcf
```

4. Merge Freebayes Results

Required

Since we ran freebayes separately on each chromosome, we need to combine each of the results into a single file for downstream processing:

```
singularity exec --bind $BIND $SIF bcftools concat -Ov $OUTDIR/freebayes_chr*.vcf >
↪$OUTDIR/freebayes.vcf
```

5. Lift hg38 to hg19

Optional

Expected Timing

< 10 min

We currently only provide reference 100G data for ancestry annotation on hg19 (GCh37).

If your sequence data was aligned to hg38 (GRCh38), you will need to lift it to hg19 for ancestry annotation with 1000G data.

We have provided chain files in the Singularity image that can be used for lifting the data between hg38 and hg19:

- `/opt/ancestry_prediction_scRNAseq/refs/GRCh38_to_GRCh37.chain` - Does not contain 'chr' enchoding (*i.e.* 1 and not chr1)
- `/opt/ancestry_prediction_scRNAseq/refs/hg38ToHg19.over.chain` - Does contain 'chr' enchoding (*i.e.* chr1 and not 1)

Define some variables to execute CrossMap to lift the data from hg38 to hg19

```
CHAIN=/opt/ancestry_prediction_scRNAseq/refs/GRCh38_to_GRCh37.chain ## Change this to␣
↪the correct chain file for your data
FASTA=/path/to/reference/fasta.fa
```

Run CrossMap to lift the data from hg19 to hg38:

```
singularity exec --bind $BIND $SIF CrossMap.py vcf $CHAIN $OUTDIR/freebayes.vcf $FASTA
↪$OUTDIR/freebayes_hg19.vcf
```

6. Convert vcf to Plink

Required

Expected Timing

< 5 min

First, convert the vcf to the plink2 pgen files. We use `max-alleles 2` because downstream softwares won't be able to deal with multi-allelic sites. The `$VCF` will be either `$OUTDIR/freebayes.vcf` (if your sequence data was mapped to hg19/GRCh37) or `$OUTDIR/freebayes_hg19.vcf` (if your sequence data was mapped to hg38/GRCh38).

```
singularity exec --bind $BIND $SIF plink2 --vcf $VCF --make-pgen --out $OUTDIR/freebayes␣
↪--max-alleles 2
```

Freebayes doesn't provide an ID for each SNP that it calls but that is important for downstream SNP filtering functions so we will create a pvar file that has IDs we will make from the chromosome, basepair, allele 1 and allele 2

```
singularity exec --bind $BIND $SIF cp $OUTDIR/freebayes.pvar $OUTDIR/freebayes.pvar_
↪original
singularity exec --bind $BIND $SIF sed -i 's/^chr//g' $OUTDIR/freebayes.pvar ### The␣
↪1000G reference that will be used doesn not have 'chr' enccoding so we will remove it␣
↪if used in your files
singularity exec --bind $BIND $SIF grep "#" $OUTDIR/freebayes.pvar > $OUTDIR/freebayes_
↪tmp.pvar
singularity exec --bind $BIND $SIF grep -v "#" $OUTDIR/freebayes.pvar | awk␣
↪'BEGIN{FS=OFS="\\t"}{print $1 FS $2 FS $1 "_" $2 "_" $4 "_" $5 FS $4 FS $5 FS $6 FS $7}'␣
↪>> $OUTDIR/freebayes_tmp.pvar
singularity exec --bind $BIND $SIF cp $OUTDIR/freebayes_tmp.pvar $OUTDIR/freebayes.pvar
```

7. Identify Common SNPs Between 1000G Data and Your Data

Required

Expected Timing

< 5 min

Next, we need to subset the variants for just those that are in common between the SNP genotypes called from freebayes and those called from the 1000G data.

Note

The 1000G data is located in the Singularity image at `/opt/1000G/all_phase3_filtered.pvar` so you can use them directly from that location as demonstrated in the below commands.

Use awk to pull SNPs that are on the same chromosomes and have the same alleles

```
singularity exec --bind $BIND $SIF awk␣
↪'BEGIN{FS=OFS="\t"}NR==FNR{a[$1,$2,$4,$5];next} ($1,$2,$4,$5) in a{print $3}' $OUTDIR/
↪freebayes.pvar /opt/1000G/all_phase3_filtered.pvar | sed '/^$/d' > $OUTDIR/common_snps/
↪snps_1000g.tsv
singularity exec --bind $BIND $SIF awk␣
↪'BEGIN{FS=OFS="\t"}NR==FNR{a[$1,$2,$4,$5];next} ($1,$2,$4,$5) in a{print $3}' /opt/
↪1000G/all_phase3_filtered.pvar $OUTDIR/freebayes.pvar | sed '/^$/d' > $OUTDIR/common_
↪snps/snps_data.tsv
```

8. Get the SNPs in Common Across all Pools

Required

Expected Timing

< 10 min

Next, we need to identify the SNPs that are in common across all the pools (and individuals if you had multiple individuals within a given pool). If you only have one pool, you will not need to do this step

Define some variables to get the common SNPs across all the pools

```
META=/path/to/metadata.tsv
```

The metadata file is the *Sample Metadata File* in the *Data Preparation* documentation.

```
singularity exec --bind $BIND $SIF Rscript /opt/ancestry_prediction_scRNAseq/scripts/
↪common_snps.R $META $OUTDIR
```

PAUSE

PAUSE

Send the resulting SNP file (`common_snps_across_pools.tsv`) to Drew Neavin at d.neavin @ garvan.org.au so that SNPs common across all sites can be used for ancestry annotation. You will need to wait until you receive the file that contains common SNPs across each site.

9. Subset 1000G Data for Common SNPs

Required

Expected Timing

< 10 min

After you have received the common SNPs file across all sites (`$COMMON_SNPS`), you can subset the 1000G and freebayes SNP data to the common SNPs. We will use `--rm-dup force-first` to help deal with possible duplicate entries for the same SNP called by freebayes.

Note

The `/opt/1000G/all_phase3_filtered` path below is the 1000G reference base filename in the `ancestry_prediction_scRNAseq.sif` singularity image. You can use the file directly from the image.

```
mkdir $OUTDIR/filter_1000g

### First need to subset the 1000g snps for the SNPs common to all sites, pools and␣
↪individuals ###
```

(continues on next page)

(continued from previous page)

```
singularity exec --bind $BIND $SIF grep -v "#" /opt/1000G/all_phase3_filtered.pvar | awk␣
↪'BEGIN{FS=OFS="\t"}{print $3}' > $OUTDIR/filter_1000g/all_1000g_snps.tsv
singularity exec --bind $BIND $SIF Rscript /opt/ancestry_prediction_scRNAseq/scripts/
↪subset_1000g_snps.R $COMMON_SNPS $OUTDIR/filter_1000g/all_1000g_snps.tsv $OUTDIR

### Subset the freebayes-called snps for the new snps ###
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile /opt/1000G/all_phase3_
↪filtered --extract $OUTDIR/snps_1000g_common_across_sites.tsv --make-pgen --out
↪$OUTDIR/filter_1000g/subset_1000g
```

10. Prune the SNPs in LD

Required

Expected Timing

< 5 min

Next, filter the SNPs for those that are not in linkage disequilibrium so we have unique representation for creating PCs:

```
singularity exec --bind $BIND $SIF plink2 --threads $N --pfile $OUTDIR/filter_1000g/
↪subset_1000g \
    --indep-pairwise 50 5 0.5 \
    --out $OUTDIR/filter_1000g/subset_pruned_1000g
```

11. Filter the 1000G and Freebayes SNPs for Pruned SNPs

Required

Expected Timing

< 5 min

We will filter the 1000G data for the SNPs in common across all sites. In addition, we'll ensure that the chromosome encoding for chromosomes X, Y and mitochondria are consistent with what is required for plink.

The only variable that needs to be defined is `$N` which is the number of threads you would like to use for this command:

```
singularity exec --bind $BIND $SIF plink2 --threads $N --pfile $OUTDIR/filter_1000g/
↪subset_1000g --extract $OUTDIR/filter_1000g/subset_pruned_1000g.prune.out --make-pgen -
↪-out $OUTDIR/filter_1000g/subset_pruned_1000g

singularity exec --bind $BIND $SIF sed -i 's/^X/23/g' $OUTDIR/filter_1000g/subset_pruned_
↪1000g.pvar
singularity exec --bind $BIND $SIF sed -i 's/^Y/24/g' $OUTDIR/filter_1000g/subset_pruned_
↪1000g.pvar
singularity exec --bind $BIND $SIF sed -i 's/^XY/25/g' $OUTDIR/filter_1000g/subset_
↪pruned_1000g.pvar
singularity exec --bind $BIND $SIF sed -i 's/^MT/26/g' $OUTDIR/filter_1000g/subset_
↪pruned_1000g.pvar
```

12. Filter Freebayes SNPs for Pruned SNPs

Required

Expected Timing

< 5 min

The only variable that needs to be defined is `$N` which is the number of threads you would like to use for this command. The `$COMMON_SNPS` is the common SNPs file across all sites:

```
singularity exec --bind $BIND $SIF plink2 --threads $N --pfile $OUTDIR/freebayes --
↪extract $COMMON_SNPS --rm-dup 'force-first' --make-pgen --out $OUTDIR/common_snps/
↪subset_data


### If have comments in the freebayes pvar file, need to transfer them
if [[ $(grep "##" $OUTDIR/common_snps/subset_data.pvar | wc -l) > 0 ]]
then
    singularity exec --bind $BIND $SIF grep "##" $OUTDIR/common_snps/subset_data.pvar >
↪$OUTDIR/common_snps/subset_pruned_data_1000g_key.txt
fi

singularity exec --bind $BIND $SIF awk -F"\\t"␣
↪'BEGIN{OFS=FS = "\\t"} NR==FNR{a[$1 FS $2 FS $4 FS $5] = $0; next} {ind = $1 FS $2 FS $4 FS $5} ind in
↪$OUTDIR/filter_1000g/subset_pruned_1000g.pvar $OUTDIR/common_snps/subset_data.pvar | \
  singularity exec --bind $BIND $SIF grep -v "##" >> $OUTDIR/common_snps/subset_pruned_
↪data_1000g_key.txt

singularity exec --bind $BIND $SIF grep -v "##" $OUTDIR/common_snps/subset_pruned_data_
↪1000g_key.txt | \
  singularity exec --bind $BIND $SIF awk 'BEGIN{FS=OFS="\t"}{print $NF}' > $OUTDIR/
↪common_snps/subset_data.prune.out

singularity exec --bind $BIND $SIF plink2 --threads $N --pfile common_snps/subset_data --
↪extract $OUTDIR/common_snps/subset_data.prune.out --rm-dup 'force-first' --make-pgen␣
↪'psam-cols='fid,parents,sex,phenos --out $OUTDIR/common_snps/subset_pruned_data

singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/subset_pruned_data.pvar
↪$OUTDIR/common_snps/subset_pruned_data_original.pvar

singularity exec --bind $BIND $SIF grep -v "#" $OUTDIR/common_snps/subset_pruned_data_
↪original.pvar | \
  singularity exec --bind $BIND $SIF awk 'BEGIN{FS=OFS="\t"}{print($3)}' > $OUTDIR/
↪common_snps/SNPs2keep.txt

singularity exec --bind $BIND $SIF grep "#CHROM" $OUTDIR/common_snps/subset_pruned_data_
↪1000g_key.txt > $OUTDIR/common_snps/subset_pruned_data.pvar

singularity exec --bind $BIND $SIF grep -Ff $OUTDIR/common_snps/SNPs2keep.txt $OUTDIR/
↪common_snps/subset_pruned_data_1000g_key.txt >> $OUTDIR/common_snps/subset_pruned_data.
↪pvar
```

(continues on next page)

(continued from previous page)

```
singularity exec --bind $BIND $SIF awk 'BEGIN{FS=OFS="\t"}NF{NF-=1};1' < $OUTDIR/common_
↪snps/subset_pruned_data.pvar > $OUTDIR/common_snps/subset_pruned_data_temp.pvar

singularity exec --bind $BIND $SIF grep "##" $OUTDIR/filter_1000g/subset_pruned_1000g.
↪pvar > $OUTDIR/common_snps/subset_pruned_data.pvar

singularity exec --bind $BIND $SIF sed -i "/^$/d" $OUTDIR/common_snps/subset_pruned_data_
↪temp.pvar

singularity exec --bind $BIND $SIF cat $OUTDIR/common_snps/subset_pruned_data_temp.pvar >
↪> $OUTDIR/common_snps/subset_pruned_data.pvar

singularity exec --bind $BIND $SIF plink2 --rm-dup 'force-first' --threads $N --pfile
↪$OUTDIR/common_snps/subset_pruned_data --make-pgen 'psam-cols='fid,parents,sex,phenos -
↪-out $OUTDIR/common_snps/final_subset_pruned_data
```

13. Update Chromosome IDs

Required

Expected Timing

< 5 min

Plink has some requirements for the chromosome IDs for SNPs on the X, Y and MT chromosomes - they need to be updated to numeric. Update the chromosome IDs to ensure compatibility with plink.

```
singularity exec --bind $BIND $SIF sed -i 's/^X/23/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.pvar
singularity exec --bind $BIND $SIF sed -i 's/^Y/24/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.pvar
singularity exec --bind $BIND $SIF sed -i 's/^XY/25/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.pvar
singularity exec --bind $BIND $SIF sed -i 's/^MT/26/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.pvar
```

14. Calculate PCs for 1000G

Required

Expected Timing

< 5 min

Next, calculate the principal components using the 1000G SNPs (alrleady filtered for the same SNPs as your dataset). Again, the only additional variable that you need to define is the number of threads (`$N`) you want to use to calculate the PCs:

```
singularity exec --bind $BIND $SIF plink2 --threads $N --pfile $OUTDIR/filter_1000g/
↪subset_pruned_1000g \
    --freq counts \
    --pca allele-wts \
    --out $OUTDIR/filter_1000g/subset_pruned_1000g_pcs
```

15. Project 1000G and Freebayes Data in PCs

Required

Expected Timing

< 5 min

Next, project the 1000G and freebayes SNPs in the PCs calculated in the last step. Again, the only additional variable that you need to define is the number of threads ($N) you want to use to calculate the PCs:

```
export OMP_NUM_THREADS=$N

singularity exec --bind $BIND $SIF plink2 --threads $N --pfile $OUTDIR/common_snps/final_
↪subset_pruned_data \
    --read-freq $OUTDIR/filter_1000g/subset_pruned_1000g_pcs.acount \
    --score $OUTDIR/filter_1000g/subset_pruned_1000g_pcs.eigenvec.allele 2 5 header-read␣
↪no-mean-imputation \
            variance-standardize \
    --score-col-nums 6-15 \
    --out $OUTDIR/pca_projection/final_subset_pruned_data_pcs

singularity exec --bind $BIND $SIF plink2 --threads $N --pfile $OUTDIR/filter_1000g/
↪subset_pruned_1000g \
    --read-freq $OUTDIR/filter_1000g/subset_pruned_1000g_pcs.acount \
    --score $OUTDIR/filter_1000g/subset_pruned_1000g_pcs.eigenvec.allele 2 5 header-read␣
↪no-mean-imputation \
            variance-standardize \
    --score-col-nums 6-15 \
    --out $OUTDIR/filter_1000g/subset_pruned_1000g_pcs_projected
```

16. Plot PC Results

Required

Expected Timing

< 5 min

Lastly, we can predict sample SNP-based ancestry and produce some figures for visualization using our wrapper script:

```
singularity exec --bind $BIND $SIF echo $OUTDIR/pca_sex_checks_original/ > $OUTDIR/pca_
↪sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR/pca_projection/final_subset_pruned_data_
↪pcs.sscore >> $OUTDIR/pca_sex_checks_original/variables.tsv
```

(continues on next page)

(continued from previous page)

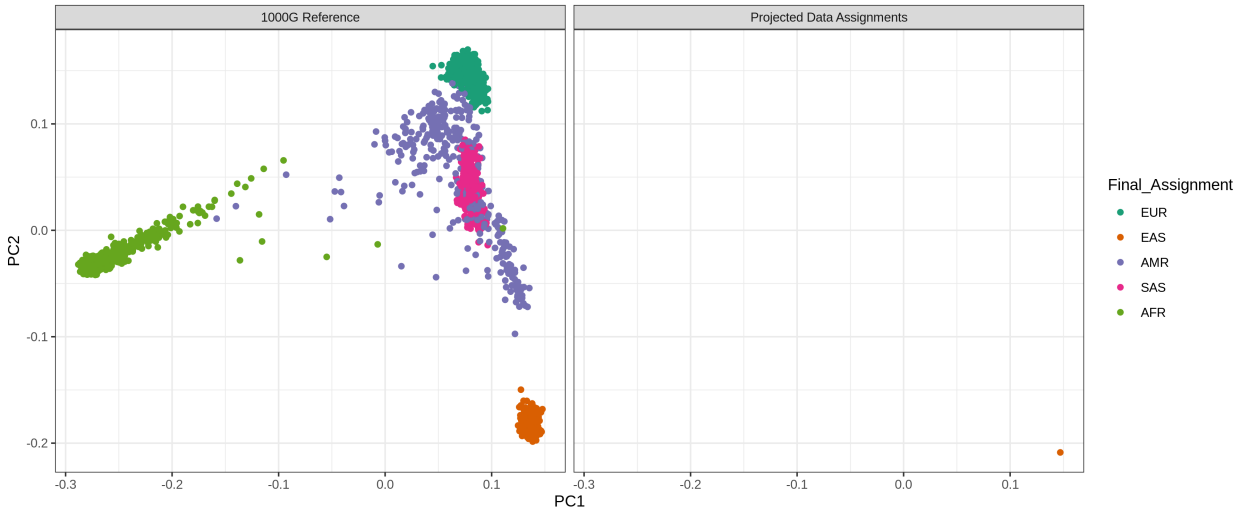
```
singularity exec --bind $BIND $SIF echo $OUTDIR/filter_1000g/subset_pruned_1000g_pcs_
↪projected.sscore >> $OUTDIR/pca_sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR/common_snps/subset_1000g.psam >> $OUTDIR/
↪pca_sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF Rscript /opt/ancestry_prediction_scRNAseq/scripts/PCA_
↪Projection_Plotting_original.R $OUTDIR/pca_sex_checks_original/variables.tsv
```

5.1.3 Results

After running the final step, you should have the following results directories. We've highlighted the key results files (`Ancestry_PCAs.png` and `ancestry_assignments.tsv`):

```
.
├── common_snps
│   ├── final_subset_pruned_data.log
│   ├── final_subset_pruned_data.pgen
│   ├── final_subset_pruned_data.psam
│   ├── final_subset_pruned_data.pvar
│   ├── snps_1000g.tsv
│   ├── SNPs2keep.txt
│   ├── snps_data.tsv
│   ├── subset_data.log
│   ├── subset_data.pgen
│   ├── subset_data.prune.out
│   ├── subset_data.psam
│   ├── subset_data.pvar
│   ├── subset_pruned_data_1000g_key.txt
│   ├── subset_pruned_data.log
│   ├── subset_pruned_data_original.pvar
│   ├── subset_pruned_data.pgen
│   ├── subset_pruned_data.psam
│   ├── subset_pruned_data.pvar
│   └── subset_pruned_data_temp.pvar
├── freebayes_hg19.vcf
├── freebayes_hg19.vcf.unmap
├── freebayes.log
├── freebayes.pgen
├── freebayes.psam
├── freebayes.pvar
├── freebayes.pvar_original
├── freebayes_tmp.pvar
├── freebayes.vcf
├── pca_projection
│   ├── final_subset_pruned_data_pcs.log
│   └── final_subset_pruned_data_pcs.sscore
└── pca_sex_checks_original
    ├── ancestry_assignments.tsv
    ├── Ancestry_PCAs.png
    └── variables.tsv
```

- The `Ancestry_PCAs.png` figure shows the 1000G individual locations in PC space compared to the individuals in each pool. For example:



- The `ancestry_assignments.tsv` file has the annotations and probabilities for each pool. For example:

FID	IID	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	AFR	AMR	EAS	EUR	SAS	Final_Assignment
0	Pool1	0.137	-0.108	-0.025	-0.042	0.032	-0.042	0.001	-0.021	-0.086	-0.019	0	0	1	0	0	EAS

5.2 Annotate Ancestry of Samples Using Reference Genotypes

We have provided instructions here to annotate your samples that have genotype data on your samples with either SNP genotype data or whole genome or exome sequencing.

This will allow users that have additional genotype data for their samples to compare the predictions from the sc sequencihng to the reference genotype data.

To do this, you will need your data will need to be on the GRCh37 (hg19) reference - sorry, we haven't provided instructions or resources for predicting ancestry for data directly on GRCh38 (hg38). So if your data is on GRCh38 (hg38), you will have to first lift your datat to hg19 (GRCh37).

The general steps are:

1. *Convert Data to Plink pgen Format* Optional
2. *Identify Common SNPs Between 1000G Data and Your Data* Required
3. *Prune the SNPs in LD* Required
4. *Filter the 1000G and Freebayes SNPs for Pruned SNPs* Required
5. *Update Chromosome IDs* Required
6. *Calculate PCs for 1000G* Required
7. *Project 1000G and Freebayes Data in PCs* Required
8. *Plot PC Results* Required
9. *Compare Reference to single cell-predicted Ancestry* Optional

5.2.1 Setup

There are a few variables that will be used throughout the example commands which are best to define in a file that you can easily run at each step or source for execution of each step.

- `$BIND` - the path(s) on your system to bind to singularity when executing commands. By default, Singularity binds just the directory (and downstream dierectory and files) from your current working directory when executing the command. However, if you have some files elsewhere on your system, you can provide a parent directory to the singulairty command to indicate which additional directories to bind. Multiple directories can be included and separated by a comma (*i.e.* $DIR1,$DIR2)
- `$SIF` - the path to the singularity image downloaded in the *Installation* section
- `$OUTDIR` - the path to the output directory where all results will be written and saved

```
BIND = /bind/path
SIF = /path/to/singularity/image/ancestry_prediction_scRNAseq.sif
OUTDIR = /path/to/base/outdir
```

5.2.2 Steps

1. Convert Data to Plink pgen Format

Optional

Expected Timing

< 10 minutes but may be longer with large datasets

This step will convert your file to a Plink pgen format which is required for input into the downstream steps. If your data is already in the Plink pgen format, you can skip this step and move directly to *Identify Common SNPs Between 1000G Data and Your Data*.

In preparation for this step, we set some additional parameters and create the required output directory. The parameters that we use for the command in this step are:

VCF File

```
VCF=/path/to/vcf.vcf
```

- This is pretty self-explanatory - this is the path to your sample genotype file.

Plink BED Files

```
BED_BASE=/path/to/plink/bed_basename
```

- The `$BED_BASE` is the your sample genotype Plink bedfiles basename. For example if your files are named `data.bed`, `data.fam` and `data.bim` your `bed_basename` would be `data`.

VCF File

```
singularity exec --bind $BIND $SIF plink2 --vcf $VCF --make-pgen --out $OUTDIR/data_name
```

Note

This command will generate three files using the out basename provided: $OUTDIR/data_name.pgen, $OUTDIR/data_name.pvar, $OUTDIR/data_name.psam

Plink BED Files

```
singularity exec --bind $BIND $SIF plink2 --bfile $BED_BASE --make-pgen --out $OUTDIR/
↪data_name
```

Note

This command will generate three files using the out basename provided: $OUTDIR/data_name.pgen, $OUTDIR/data_name.pvar, $OUTDIR/data_name.psam

2. Identify Common SNPs Between 1000G Data and Your Data

Required

This step will identify the genetic variants in common between your data and the 1000G reference and then subset your data and 1000G reference data for the genetic variants in common between the two datasets. The 1000G reference data is provided within the Singularity image so you will not have to download it separately. The paths to the 1000G reference data are the paths within the Singularity image.

```
singularity exec --bind $BIND $SIF awk␣
↪'NR==FNR{a[$1,$2,$4,$5];next} ($1,$2,$4,$5) in a{print $3}' $OUTDIR/data_name.pvar /
↪opt/1000G/all_phase3_filtered.pvar > $OUTDIR/common_snps/snps_1000g.tsv
singularity exec --bind $BIND $SIF awk␣
↪'NR==FNR{a[$1,$2,$4,$5];next} ($1,$2,$4,$5) in a{print $3}' /opt/1000G/all_phase3_
↪filtered.pvar $OUTDIR/data_name.pvar > $OUTDIR/common_snps/snps_data.tsv

singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile $OUTDIR/data_name --
↪extract $OUTDIR/common_snps/snps_data.tsv --make-pgen 'psam-cols='fid,parents,sex,
↪phenos --out $OUTDIR/common_snps/subset_data
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile /opt/1000G/all_phase3_
↪filtered --extract $OUTDIR/common_snps/snps_1000g.tsv --make-pgen --out $OUTDIR/common_
↪snps/subset_1000g
```

3. Prune the SNPs in LD

Required

This step will prune the genotype data so that the genetic variants that are in low linkage disequilibrium.

```
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile /opt/1000G/all_phase3_
↪filtered \
    --indep-pairwise 50 5 0.5 \
    --out $OUTDIR/common_snps/subset_pruned_1000g
```

4. Filter the 1000G and Freebayes SNPs for Pruned SNPs

Required

This step will prune the genotype data so that the genetic variants that are in low linkage disequilibrium (uniquely representing genetic variation across the genome).

```
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile /opt/1000G/all_phase3_
↪filtered --extract $OUTDIR/common_snps/subset_pruned_1000g.prune.out --make-pgen --out
↪$OUTDIR/common_snps/subset_pruned_1000g

if [[ $(grep "##" $OUTDIR/common_snps/subset_1000g.pvar | wc -l) > 0 ]]
then
    singularity exec --bind $BIND $SIF grep "##" $OUTDIR/common_snps/subset_1000g.pvar >
↪$OUTDIR/common_snps/subset_pruned_data_1000g_key.txt
fi

singularity exec --bind $BIND $SIF awk -F"\\t"␣
↪'BEGIN{OFS=FS = "\\t"} NR==FNR{a[$1 FS $2 FS $4 FS $5] = $0; next} {ind = $1 FS $2 FS $4 FS $5} ind in
↪$OUTDIR/common_snps/subset_pruned_1000g.pvar $OUTDIR/common_snps/subset_pruned_data.
↪pvar | singularity exec --bind $BIND $SIF grep -v "##" >> $OUTDIR/common_snps/subset_
↪pruned_data_1000g_key.txt
singularity exec --bind $BIND $SIF grep -v "##" $OUTDIR/common_snps/subset_pruned_data_
↪1000g_key.txt | singularity exec --bind $BIND $SIF awk 'BEGIN{FS=OFS="\t"}{print $NF}'␣
↪> $OUTDIR/common_snps/subset_data.prune.out
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile $OUTDIR/common_snps/subset_
↪data --extract $OUTDIR/common_snps/subset_data.prune.out --make-pgen 'psam-cols='fid,
↪parents,sex,phenos --out $OUTDIR/common_snps/subset_pruned_data
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/subset_pruned_data.pvar
↪$OUTDIR/common_snps/subset_pruned_data_original.pvar
singularity exec --bind $BIND $SIF grep -v "#" $OUTDIR/common_snps/subset_pruned_data_
↪original.pvar | singularity exec --bind $BIND $SIF awk 'BEGIN{FS=OFS="\t"}{print($3)}'␣
↪> $OUTDIR/common_snps/SNPs2keep.txt
singularity exec --bind $BIND $SIF grep "#CHROM" $OUTDIR/common_snps/subset_pruned_data_
↪1000g_key.txt > $OUTDIR/common_snps/subset_pruned_data.pvar
singularity exec --bind $BIND $SIF grep -Ff $OUTDIR/common_snps/SNPs2keep.txt $OUTDIR/
↪common_snps/subset_pruned_data_1000g_key.txt >> $OUTDIR/common_snps/subset_pruned_data.
↪pvar
singularity exec --bind $BIND $SIF awk 'BEGIN{FS=OFS="\t"}NF{NF-=1};1' < $OUTDIR/common_
↪snps/subset_pruned_data.pvar > $OUTDIR/common_snps/subset_pruned_data_temp.pvar
singularity exec --bind $BIND $SIF grep "##" $OUTDIR/common_snps/subset_pruned_data_temp.
↪pvar > $OUTDIR/common_snps/subset_pruned_data.pvar
```

(continues on next page)

(continued from previous page)

```
singularity exec --bind $BIND $SIF cat $OUTDIR/common_snps/subset_pruned_data_temp.pvar >
↪> $OUTDIR/common_snps/subset_pruned_data.pvar
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile $OUTDIR/common_snps/subset_
↪pruned_1000g --make-bed --out $OUTDIR/common_snps/subset_pruned_1000g

### This is a contingency to remove duplicated snps from both
singularity exec --bind $BIND $SIF plink2 --rm-dup 'force-first' -threads 2 --pfile
↪$OUTDIR/common_snps/subset_data --make-pgen 'psam-cols='fid,parents,sex,phenos --out
↪$OUTDIR/common_snps/final_subset_pruned_data
singularity exec --bind $BIND $SIF plink2 --rm-dup 'force-first' -threads 2 --pfile
↪$OUTDIR/common_snps/subset_data --make-bed --out $OUTDIR/common_snps/subset_pruned_data
```

5. Update Chromosome IDs

Required

Plink requires specific chromosome encodings for chromosomes X, Y, and mitochondria so we will update them to be sure.

```
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/final_subset_pruned_data.bed
↪$OUTDIR/common_snps/split/
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/final_subset_pruned_data.bim
↪$OUTDIR/common_snps/split/
singularity exec --bind $BIND $SIF sed -i 's/^X/23/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.bim
singularity exec --bind $BIND $SIF sed -i 's/^Y/24/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.bim
singularity exec --bind $BIND $SIF sed -i 's/^XY/25/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.bim
singularity exec --bind $BIND $SIF sed -i 's/^MT/26/g' $OUTDIR/common_snps/split/final_
↪subset_pruned_data.bim
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/final_subset_pruned_data.fam
↪$OUTDIR/common_snps/split/
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/subset_pruned_1000g.bed
↪$OUTDIR/common_snps/split/
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/subset_pruned_1000g.bim
↪$OUTDIR/common_snps/split/
singularity exec --bind $BIND $SIF cp $OUTDIR/common_snps/subset_pruned_1000g.fam
↪$OUTDIR/common_snps/split/
singularity exec --bind $BIND $SIF sed -i 's/^X/23/g' $OUTDIR/common_snps/split/subset_
↪pruned_1000g.bim
singularity exec --bind $BIND $SIF sed -i 's/^Y/24/g' $OUTDIR/common_snps/split/subset_
↪pruned_1000g.bim
singularity exec --bind $BIND $SIF sed -i 's/^XY/25/g' $OUTDIR/common_snps/split/subset_
↪pruned_1000g.bim
singularity exec --bind $BIND $SIF sed -i 's/^MT/26/g' $OUTDIR/common_snps/split/subset_
↪pruned_1000g.bim
```

6. Calculate PCs for 1000G

Required

Next, we will calculate principal components using the 1000G reference data.

```
singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile $OUTDIR/common_snps/subset_
↪pruned_1000g \
    --freq counts \
    --pca allele-wts \
    --out $OUTDIR/pca_projection/subset_pruned_1000g_pcs
```

7. Project 1000G and Freebayes Data in PCs

Required

Next, let's project the data (both your data and the 1000G reference data) into the principal component space that you calculated in the last step.

```
export OMP_NUM_THREADS=2

singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile $OUTDIR/common_snps/final_
↪subset_pruned_data \
    --read-freq $OUTDIR/pca_projection/subset_pruned_1000g_pcs.acount \
    --score $OUTDIR/pca_projection/subset_pruned_1000g_pcs.eigenvec.allele 2 5 header-
↪read no-mean-imputation \
            variance-standardize \
    --score-col-nums 6-15 \
    --out $OUTIR/pca_projection/final_subset_pruned_data_pcs

singularity exec --bind $BIND $SIF plink2 --threads 2 --pfile {params.infile_1000g} \
    --read-freq $OUTDIR/pca_projection/subset_pruned_1000g_pcs.acount \
    --score $OUTDIR/pca_projection/subset_pruned_1000g_pcs.eigenvec.allele 2 5 header-
↪read no-mean-imputation \
            variance-standardize \
    --score-col-nums 6-15 \
    --out $OUTDIR/pca_projection/subset_pruned_1000g_pcs_projected
```

8. Plot PC Results

Required

Fnally, let's plot the results and predict the ancestry of the samples in your dataset.

```
singularity exec --bind $BIND $SIF echo $OUTDIR/pca_sex_checks_original > $OUTDIR/pca_
↪sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR/pca_projection/final_subset_pruned_data_
↪pcs.sscore >> $OUTDIR/pca_sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR/pca_projection/subset_pruned_1000g_pcs_
↪projected.sscore >> $OUTDIR/pca_sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR/common_snps/subset_1000g.psam >> $OUTDIR/
↪pca_sex_checks_original/variables.tsv
singularity exec --bind $BIND $SIF Rscript /opt/ancestry_prediction_scRNAseq/scripts/PCA_
↪Projection_Plotting_original.R $OUTDIR/pca_sex_checks_original/variables.tsv
```

(continues on next page)

(continued from previous page)

9. Compare Reference to single cell-predicted Ancestry

Optional

The last step is to compare the predicted ancestries in your genotyped data (either microarray or whole genome or exome sequencing data) to the predicted ancestries from your single cell data. We assume that the ID you use for the sample is the same in bpoth the single cell and genotyped data.

You will need to provide a metadata file for each of the Pools and samples in each pool as input for this step. The file is the same *Sample metadata file*.

In preparation for this step, we set some additional parameters and create the required output directory. The parameters that we use for the command in this step are:

```
META=/path/to/meta_file.tsv
```

We assume that the results for each of the samples that you have single cell sequencing are located in the same base directory in the following format:

```
.
├── Pool1
│   ├── individual_1
│   ├── ...
│   └── individual_n
│   ...
└── Poolm
    ├── individual_1
    ├── ...
    └── individual_n
```

```
singularity exec --bind $BIND $SIF echo $OUTDIR > $OUTDIR/ref_sc_ancestry_prediction_
↪comparison/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR/pca_sex_checks_original/ancestry_
↪assignments.tsv >> $OUTDIR/ref_sc_ancestry_prediction_comparison/variables.tsv
singularity exec --bind $BIND $SIF echo $OUTDIR >> $OUTDIR/ref_sc_ancestry_prediction_
↪comparison/variables.tsv
singularity exec --bind $BIND $SIF echo $META >> $OUTDIR/ref_sc_ancestry_prediction_
↪comparison/variables.tsv
singularity exec --bind $BIND $SIF Rscript /opt/ancestry_prediction_scRNAseq/scripts/
↪compare_ref_seq_snp_ancestries.R $OUTDIR/ref_sc_ancestry_prediction_comparison/
↪variables.tsv
```

5.2.3 Results

After running the final step, you should have the following results directories.

We've highlighted the key results files (`Ancestry_PCAs.png` and `ancestry_assignments.tsv`):

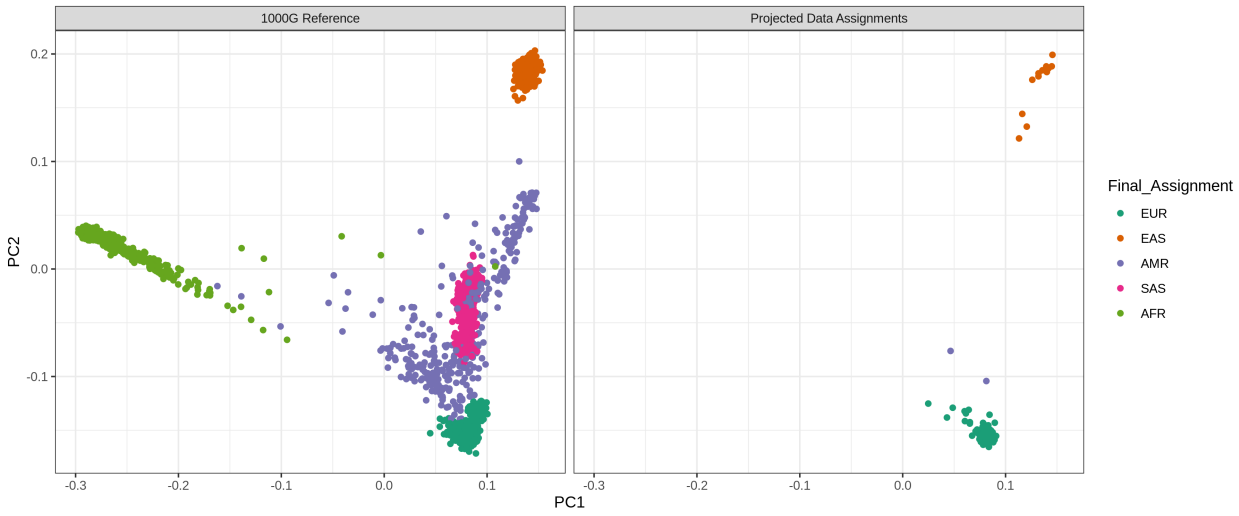
```
.
├── common_snps
│   ├── final_subset_pruned_data.bed
│   ├── final_subset_pruned_data.bim
│   ├── final_subset_pruned_data.fam
│   ├── final_subset_pruned_data.log
│   ├── final_subset_pruned_data.pgen
│   ├── final_subset_pruned_data.psam
│   ├── final_subset_pruned_data.pvar
│   ├── snps_1000g.tsv
│   ├── SNPs2keep.txt
│   ├── snps_data.tsv
│   ├── subset_1000g.log
│   ├── subset_1000g.pgen
│   ├── subset_1000g.psam
│   ├── subset_1000g.pvar
│   ├── subset_data.log
│   ├── subset_data.pgen
│   ├── subset_data.prune.out
│   ├── subset_data.psam
│   ├── subset_data.pvar
│   ├── subset_pruned_1000g.bed
│   ├── subset_pruned_1000g.bim
│   ├── subset_pruned_1000g.fam
│   ├── subset_pruned_1000g.log
│   ├── subset_pruned_1000g.pgen
│   ├── subset_pruned_1000g.popu
│   ├── subset_pruned_1000g.prune.in
│   ├── subset_pruned_1000g.prune.out
│   ├── subset_pruned_1000g.psam
│   ├── subset_pruned_1000g.pvar
│   ├── subset_pruned_data_1000g_key.txt
│   ├── subset_pruned_data.log
│   ├── subset_pruned_data_original.pvar
│   ├── subset_pruned_data.pgen
│   ├── subset_pruned_data.psam
│   ├── subset_pruned_data.pvar
│   └── subset_pruned_data_temp.pvar
├── pca_projection
│   ├── final_subset_pruned_data_pcs.log
│   ├── final_subset_pruned_data_pcs.sscore
│   ├── subset_pruned_1000g_pcs.acount
│   ├── subset_pruned_1000g_pcs.eigenval
│   ├── subset_pruned_1000g_pcs.eigenvec
│   ├── subset_pruned_1000g_pcs.eigenvec.allele
│   ├── subset_pruned_1000g_pcs.log
│   ├── subset_pruned_1000g_pcs_projected.log
│   └── subset_pruned_1000g_pcs_projected.sscore
```

(continues on next page)

(continued from previous page)

```
├── pca_sex_checks_original
│   ├── ancestry_assignments.tsv
│   ├── Ancestry_PCAs.png
│   └── variables.tsv
├── reference.log
├── reference.pgen
├── reference.psam
└── reference.pvar
```

- The `Ancestry_PCAs.png` figure shows the 1000G individual locations in PC space compared to the individuals in each pool. For example:



- The `ref_ancestry_assignments.tsv` file has the annotations and probabilities for each pool. For example:

FID	IID	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	AFR	AMR	EAS	EUR	SAS	combined_assignment	Final_Assignment
0	1	0.137	-0.108	-0.025	-0.042	0.032	-0.042	0.001	-0.021	-0.086	-0.019	0	0	1	0	0	EAS	EAS
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

5.3 Support

If you have any questions, suggestions or issues with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue or email Drew Neavin (d.neavin @ garvan.org.au)

CHAPTER

SIX

CONTACT

This ancestry prediction pipeline has been developed by Dr Drew Neavin in Joseph Powell's Lab at the Garvan Institute of Medical Research.

You can contact us with questions, issues or recommendations with a Github issue.

CHAPTER

SEVEN

SUPPORT

If you're having trouble with any part of the Ancestry Prediction from scRNA-seq Data Pipeline, feel free to submit an issue.