PoolParty Pipeline Example

by: Steven Micheletti

Synopsis:

This exercise uses 150 PE fastq files from a fake diploid organism (found in the 'example' folder) to illustrate performance of the three modules of *PoolParty*. If interested in using *PoolParty*, this is a good starting point to ensure the pipeline will run on your system and to better understand output files.

We will use a hypothetical example to detect genomic differences between category A with phenotype "X" and category B with phenotype "Z." Specifically, we will be aligning 5 libraries (10 *fastq* files) to a small (9.3 mb) reference genome (*PP_genome.fa*)

PP_genome.fa	-Small genome with 3 anchored chromosomes and 5 scaffolds
P101_103_R1_PopA.fq.gz	-Individual 101, read 1, from pop "A"
P101_103_R2_PopA.fq.gz	-Individual 101, read 2, from pop "A"
P102_97_R1_PopA.fq.gz	-Individual 102, read 1, from pop "A"
P102_97_R2_PopA.fq.gz	-Individual 102, read 2, from pop "A"
P201_104_R1_PopB.fq.gz	-Individual 201, read 1, from pop "B"
P201_104_R2_PopB.fq.gz	-Individual 201, read 2, from pop "B"
P202_103_R1_PopB.fq.gz	-Individual 202, read 1, from pop "B"
P202_103_R2_PopB.fq.gz	-Individual 202, read 2, from pop "B"
P203_97_R1_PopB.fq.gz	-Individual 203, read 1, from pop "B"
P203_97_R2_PopB.fq.gz	-Individual 203, read 2, from pop "B"

Note that the content of the *fastq* files have been greatly reduced for example purposes. Therefore, we will be interpreting regions with very low coverage.

1) Genome Preparation

In some cases, a reference genome may have been previously indexed and prepared. Whenever acquiring a new reference genome *fasta*, it is essential to first prepare the genome.

prep_genome.sh contains the commands needed to prepare *PP_genome.fa*. You may either run this as a bash script or run the commands independently:

First, we index the genome with *bwa*. This allows *bwa mem* to align reads to the genome:

```
> bwa index -a bwtsw PP_genome.fa
```

Next, we index the genome with *samtools*. This produces a *.fai* file with useful information about each chromosome/scaffold's size:

```
> samtools faidx PP_genome.fa
```

Finally, we create a dictionary with *Picard Tools*. This allows *Picard* to properly sort aligned reads:

```
> -jar picard.jar CreateSequenceDictionary \
                  REFERENCE= PP_genome.fa \
                  OUTPUT= PP_genome.fa.dict
```

After running these commands a number of files with additional extensions will be produced: *.sa, .fai., .dict, .pac, .bwt, .ann, and .amb*. It is important that genome name serves as a prefix to these the files (*e.g., PP_genome.fa.dict*).

2) PPalign

It is assumed that at this point that all dependencies have been installed. *PPalign* will report an error if it is unable to load any dependency. Some R packages are also required and will be attempted to be installed automatically if not detected on the system.

a) *Creating samplelist.txt*

In this example, samplelist.txt has already been created. This file contains one *fastq* file per line with its corresponding category. In this example, we have 5 libraries that belong two different categories. A category can be known population assignment or a known trait. For this example, let's say that category A has phenotype "X" and category B has phenotype "Z".

Open *samplelist.txt*:

> *samplelist.txt*

```
P101_103_R1_PopA.fq.gz      1
P101_103_R2_PopA.fq.gz      1
P102_97_R1_PopA.fq.gz       1
P102_97_R2_PopA.fq.gz       1
P201_104_R1_PopB.fq.gz      2
P201_104_R2_PopB.fq.gz      2
P202_103_R1_PopB.fq.gz      2
P202_103_R2_PopB.fq.gz      2
P203_97_R1_PopB.fq.gz       2
P203_97_R2_PopB.fq.gz       2
```

Naming convention of the *fastq* files is somewhat flexible but has some requirements. The prefix, or text before the first underscore, must be a unique identifier that is shared between paired libraries. In this example, we have two *fastqs* for each unique library 101, 102 ,201 ,202 , and 203. In this example, after the unique ID, an optional field that denotes lane information is present. Next, a required field denoting if the file is read 1 (R1) or read 2 (R2) is included to differentiate the paired-end reads. Finally, an optional population field is included at the end of the file name, before the extension (PopA/PopB). The extension of the file does not matter and could follow common *fastq* conventions such as *.fastq, fq., fastq.gz, etc.*

Next to each filename, *PPalign* will recognize shared category assignment (1 or 2 in this case, which corresponds to phenotype "X" and "Z") and merge these alignments at a later step. Category assignment must be an integer and should range from 1-number of categories.

More examples of file names that will work:

```
Pop1_FR.fq 1
Pop1_RR.fq 1
452_202i_1.fastq 2
452_202i_2.fastq 2
```

Examples of file names that WILL NOT work:

```
603_ind1_R1.fq 1
603_ind1_R2.fq 1
603_ind2_R1.fq 2
603_ind2_R2.fq 2
```

In this example, PPalign sees only one individual (603). The script will exit with an error.

POP1_R1.fastq 1
POP1_102_R2.fastq 1

In this example, *PPalign* does not think these are matching reads because read 2 has excess information that is not seen in read 1. The script will exit with an error.

b) Editing the config file.

The configuration file contains directory locations, parameters, and dependency locations. You will need to edit *pp_align.config* to tailor directories to your system.

Note, that there cannot be any spaces between "=" and the definition of the variable. The script will exit with an error if so.

Open *pp_align.config*

> *pp_align.config*

#Input/Output#

INDIR= /usr/local/bin/poolparty/example

Change this to the directory where the example *fastq* files and *samplelist.txt* are.

OUTDIR= /userdir/phenoAB

Change this to a directory where you have permission to write files to.

OUTPOP=phenoAB

You may leave this as is or change it to a more familiar name. This will be the prefix to any file that is written.

GENOME=/usr/local/bin/poolparty/example/PP_genome.fa

Specify the genome name with its path.

SCAHEAD=Scaff

Leave this as is. If you open *PP_genome.fa.fai* you will see the three chromosomes and their lengths in basepairs. In this reference genome, there are 5 scaffolds are noted by Scaff. By indicating the scaffold heading, we are telling *PPalign* how to differentiate chromosomes from scaffolds. This is optional.

#Run Parameters#*

THREADZ=4

Some packages can be processed in parallel through multi-threading. Specific the maximum number of threads to use at a time. 4 should be more than enough for this example.

BQUAL=20

Base quality refers to PHRED score given to each base in a read. Base quality tends to deteriorate towards the ends of reads and low-quality bases should be trimmed to reduce inaccurate base calls. *BBduk* from *BBmap* uses this quality to remove a read if its average base quality falls below this value, or trim a region of reads if their average quality is below this value.

MAPQ=5

Mapping quality is defined differently by various aligners. In general, it is a score denoting how well reads have aligned to the reference genome. Reads are penalized by factors such as mismatches. A MAPQ score of 0 means that reads are aligning to different portions of the genome with the same probability. With lower MAPQ, more reads will be retained but have lower confidence in alignment calls; higher MAPQ will retain fewer reads, but with higher confidence. For this example, we will use a low MAPQ of 5.

SNPQ=20

SNP quality (QUAL) will depend on the package calling SNPs. In this case, *bcftools* will provide a QUAL which is confidence in that the genomic position is indeed a SNP. For

instance, a putative heterozygous position may be a variant, or perhaps all individuals are fixed for the heterozygous genotype. *bcftools* will take into account read depth at the reference and alternate alleles for all libraries and provide QUAL score. The main purpose of this filter in *PPalign* is to reduce file sizes and thus memory usage. A SNPQ of 20 removes highly unlikely SNPs; however, this value may want to be raised or lowered based on the purpose of the experiment. For this example, we will use SNPQ=20.

MINLENGTH=25

After bases are trimmed by quality, the read may become so short that it cannot properly align to the reference genome. MINLENGTH is the minimum length a read must be after trimming to retain it. Base-ically, MINLENGTH=25 means discard any reads that fall below 25 bp after quality trimming.

INWIN=15

Alignment error can occur around indels leading to false-positive SNPs being called in these regions. INWIN is the number of basepairs around an indel to discard when calling SNPs. This is a quick alternative to local realignment. For this example, let's keep INWIN at 15, which is a large region for indel-region SNP removal.

MAF=0.05

Global minor allele frequency. Removes SNP calls where the minor allele is below MAF. Will greatly reduce the number of uninformative SNPS and thus file sizes.

KMEM=Xmx4g

Memory parameter for java-based programs. Memory issues with java can be common and this parameter may need to be tailored to your specific system. Xmx means the maximum size in the memory allocation pool, with 4g being 4gb. If unsure about optimal memory parameters leave as is.

MINDP=10

The minimum depth of coverage across all libraries required to retain a SNP. Depth of coverage filters are applied during the analysis step as well; this serves as an initial screening step to reduce the number of SNPs and thus file sizes. For this example, we will use a MINDP of 10.

#Run-types#

SPLITDISC=off

Samblaster can identify split-end and discordant aligned reads and output them as separate *sam* files. These read types are useful when investigating structure variants through atypical alignment of reads. SPLITDISC=off will skip this production which can save computation time and space. Leave off for example.

INDCONT=on

Pooled sequencing has traditionally consisted of sequencing a pool of mixed DNA as one. However, recent library protocols have incorporated a barcoding step which allows identification of individual reads. If individuals were barcoded, INCONT=on will run scripts that check for variance in allele contribution of SNPs and produce individual stats. Additionally, a normalization process will occur whereby each individual's allelic contribution to a SNP is normalized. Note that this process is memory intensive. Leave on for example.

QUALREPORT=on

fastqc provides helpful quality plots and stats on trimmed reads. Sometimes quality reports are not needed and QUALREPORT=off will prevent *fastqc* from running. Leave on for example

#Dependency Locations#

BCFTOOLS=bcftools
FASTQC=fastqc
BWA=bwa
SAMBLASTER=samblaster
SAMTOOLS=samtools
PICARDTOOLS=/usr/local/bin/picard.jar

BBMAPDIR=/usr/local/bin/bbmap/
POOL2=/usr/local/bin/popoolation2_1201/

PPalign will ensure that all dependencies are installed and throw an error if not. Ensure that the correct command and/or directory is set for each dependency. Note that R, perl, and java languages are required as well.

*Note on additional parameters: If knowledgeable about the packages used in *PPalign*, additional parameters can be included to tailor analyses towards specific study organisms. To add additional parameters for *bwa*, *samtools*, *bcftools*, and *bbmap* arguments can simply be added in *PPalign.sh* after each command. A comment "##! ADDITIONAL PARAMETERS…" in the script will indicate where these additional arguments can be added.

c) Running PPalign

At this point we have a study question, a samplelist, and a config file ready to go. We are now ready to run *PPalign*.

If you have administrative privileges, it is convenient to run *PPalign* directly from the command line (without specifying the directory). To do this, create a symbolic link in an installation folder to *PPalign.sh*:

```
>  ln -s /usr/local/bin/poolparty/PPalign.sh  /usr/local/bin/PPalign
```

To run *PPalign*, we simply pass the config file as an argument. Make sure you are in the directory where the config file is located and run *PPalign :*

```
>  PPalign pp_align.config &> example_log.out &
```

This will write all alerts to *example_log.out* in the current directory, which contains a lot of useful information. You may also run the command without "&> example_log.out &" but alerts will not be saved to a file.

d) PPalign Results

If executed properly, thousands of lines of alerts will be written to the log file and output files will be written to the output directory. The whole alignment module should take <5 mins to finish in this example.

A copy of your config file with the run date is produced in the output folder. This allows you to continue to modify a single config file for different *PPalign* runs since the parameters for each run will be saved in the respective output folder.

Let's go through what is happening step by step:

i) The log file.

Using Unix, you can quickly query PoolParty-specific Alerts in the log file (*example_log.out*)

```
> grep "ALERT " example_log.out
```

The status of the run should be produced. Most analyses have a timestamp that gives an indication on how fast each portion of the module run.

If you suspect errors:

```
> grep "ERROR" example_log.out
```

If you are interested in the number of optical PCR duplicates produced:

```
> grep "duplicates" example_log.out
```

When you query the log file, the order of *fastqs* will be consistent. You can check on how the pipeline ordered names in the output folder:

> OUTDIR/*phenoAB_names.txt*

P101_103_R1_PopA
P102_97_R1_PopA
P201_104_R1_PopB
P202_103_R1_PopB
P203_97_R1_PopB

This is the order that the *fastqs* are processed and combined for relevant downstream analyses.

> OUTDIR/pops/*pop_1.txt*

Shows you which libraries belong to population 1.

ii) BBduk quality trimming.

BBduk performs the first step of the pipeline by quality trimming, kmer trimming, and contaminant (adapters, dimer, *etc.*) removal.

```
> grep "Low quality" example_log.out
```

Describes the number low-quality reads and bases removed :

```
Low quality discards:   3344 reads (3.60%)   339351 bases (2.57%)
Low quality discards:   3006 reads (2.56%)   305700 bases (1.82%)
Low quality discards:   1660 reads (2.41%)   152345 bases (1.57%)
Low quality discards:   1564 reads (2.32%)   139202 bases (1.47%)
Low quality discards:   1052 reads (1.72%)   86567 bases (1.01%)
```

```
> grep "Contaminant" example_log.out
```

Indicates that no contaminants were detected. However, real sequencing data will likely have a proportion of contamination from adapters and primer dimers. *BBMap* contains a contaminant list in /resources/*adapters.fa* , which contains common dimer and adapter sequences. This list is fully customizable and you may add contaminant sequences based on the library preparation protocol you are using. For instance, if you are using a NebNext kit, you may want to look up NebNext primers and include them in this list.

We can see more information about trimmed reads in OUTPUT/trimmed/ , which contains the trimmed *fastqs* and stats for each. The trim stats have some basic information, like number of reads, but will also produce the source of contamination if any is detected.

iii) Fastqc quality checking

Fastqc in *PoolParty* checks for quality AFTER trimming, meaning that quality reports should show high-quality reads. OUTPUT/quality/ has html files with summary stats on each read.

If we open *P203_97_R1_PopB.trim_2_fastqc.html* we check summary of the quality:

Measure	Value
Filename	P203_97_R1_PopB.trim_2
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	27761
Sequences flagged as poor quality	0
Sequence length	25-151
%GC	42

In some cases, we may want to tune the BQUAL parameter to allow for more data, or filter out more low-quality reads.

iv) Read Alignment and bam filtering

Trimmed reads are aligned to the reference genome with *bwa mem* while duplicates are removed by *samblaster* and mapping quality is filtered by *samtools*.

Aligned bam files are then processed in three steps:

1) *aligned.bam* = trimmed reads aligned to the specified reference genome
   bams that have been aligned by *bwa mem*, filtered by MAPQ, and have PCR duplicates removed.
2) *sorted.bam* = aligned bam files that have been coordinate-sorted by *Picard*
   bams sorted by chromosome and position coordinates.
3) *filtered.bam* = sorted bam files that have been filtered by *samtools*
   bams with uninformative read information removed. Unpaired and unaligned reads are removed to reduce size of the *bam* files.

Note that after the filtered *bam* files are created, the aligned *bam* files are deleted to save space. Therefore, if we go into /OUTDIR/BAM, we should only see sorted and filtered *bam* files.

After the *bam* files are aligned, *samtools* stat reports are also generated for each. These can be found in OUTDIR/reports. Let's look at one:

> *P203_97_R1_PopB_phenoAB_aln_report.txt*

```
43119 + 0 in total (QC-passed reads + QC-failed reads)
506 + 0 secondary
0 + 0 supplementary
0 + 0 duplicates
43119 + 0 mapped (100.00% : N/A)
42613 + 0 paired in sequencing
21285 + 0 read1
21328 + 0 read2
41263 + 0 properly paired (96.83% : N/A)
42172 + 0 with itself and mate mapped
441 + 0 singletons (1.03% : N/A)
105 + 0 with mate mapped to a different chr
105 + 0 with mate mapped to a different chr (mapQ>=5)
```

The total number of reads (43119) refers to the total number of reads *after* quality trimming. Secondary reads (506) are those that have a high chance of aligning to multiple positions in the ref genome. We should see 100% mapped (43119) every time since the report is performed on a bam that has been aligned to ref genome. However, we are generally interested in properly paired (41263) as these are the reads that have aligned with their mate in proper orientation and will be retained for SNP calling.

We also see *pop_1.bam* and *pop_2.bam*, which are merged *bam* files. All aligned reads with phenotype "X" are in *pop_1.bam*, and all aligned reads with phenotype "Z" are in *pop_2.bam*. In other words, *fastq* files with the same categorical designation (1 or 2 in this case) in *samplelist.txt* will be combined.

v) SNP calling

It can be a waste of computational power and space to process and store all aligned genomic positions (*i.e.,* monomorphic sites). Therefore, *PPalign* first uses *bcftools* to call all potential SNPs, then uses these SNPs to filter other file formats, essentially treating the SNP calls as whitelist.

bcftools uses the combined *bam* files (*pop_1.bam* and *pop_2.bam*) to predict all potential SNPs based on all individual libraries and creates a variant call format (VCF) file. Then, *bcftools* uses user inputted parameters to filter out unlikely and rare SNPs with the SNP quality (SNPQ), global minor allele frequency (MAF), and total depth of coverage (MIDP) filters specified in the config file. This not only reduces space, but will speed up downstream analyses.

In our example we can see how many SNPS were called:

```
> grep "SNPs" example_log.out
```

```
ALERT: Samtools and bcftools are calling SNPs and creating mpileup at Thu Mar  8 15:57:17 MST 2018
ALERT: 217546 SNPs total SNPS called without filters
ALERT: 202716 SNPs removed due to QUAL < 20 and total DP < 10
ALERT: Additional 905 SNPs removed due to global MAF < 0.05
ALERT: 13925 total SNPs retained after SNP calling
ALERT: Of the remaining SNPs, there are 13731 SNPs and 194 INDels
```

An original VCF* file with 217546 SNPs was produced; however, the majority of these SNPs were filtered out due to the various filters, leaving only 13,925 SNPs. We additionally see that of these SNPs, 194 are indels. *PPalign* will split the variants calls into two files in OURDIR/filters, one listing all indels and the other listing all SNPs.

PPalign creates a mpileup file (*phenoAB.mpileup*) while simultaneously filtering it by the SNP positions above. This *mpileup* file is used for making a *sync* format that is required for downstream comparative analyses.

Another large *mpileup* is created for all aligned positions in each bam. This is generally a very large file , though quality information is stripped to reduce size (*phenoAB_stats.mpileup*). This file is coverage information for each category (pop_1 or pop_2) at each aligned genomic position and is required for *PPstats*. If stats aren't needed and storage is an issue, this file can be deleted.

phenoAB_variants.txt shows the reference chromosome and position for all the remaining SNPs with SNP quality and depth. Additionally, indels are noted with "IDV." This file allows quick and intuitive access to the SNPs you will analyzing.

*Note that if individuals are barcoded, it is possible to use the VCF file to call individual genotypes. Using *bcftools* (or called *vcftools*), you can filter individual sites that have enough coverage (usually 30X minimum) to call individual genotypes with high confidence.

vi) Sync creation

With the SNP-filtered *mpileup*, *PPalign* creates a *sync* format, which is a *Popoolation*-based format. This intuitive format has a column for each library with six digits separated by colons:

```
> head -3 phenoAB.sync
```

```
Chr1  6722   G   1:0:0:4:0:0   1:0:0:0:0:0
Chr1  10101  C   0:1:1:0:0:0   0:3:3:0:0:0
Chr1  10104  T   0:1:1:0:0:0   0:3:3:0:0:0
```

The first column is the reference chromosome or scaffold name, the second is the position within the chromosome, and the third is the reference allele. Additional columns refer to allele counts for pop_1 and pop_2 whereby the order of the positions refer to bases: A T C G N del

For example, the first line of *phenoAB.sync* denotes that pop_1 has 1 copy of the A allele and 4 of the G allele, and pop_2 only has one copy of the A allele. *phenoAB.sync*

Popoolation2 uses the *mpileup* file to identify indels and remove SNPs that fall within our specified INWIN of 15 bp. This generally means a range of about 30 bp will be marked around indels and removed. This information is shown in phenoAB.gtffile

```
> head -3  phenoAB.gtffile
```

```
Chr1  mpileup indelregion  5      34     .  .  .  gene_id "."; transcript_id ".";
Chr1  mpileup indelregion  10140  10181  .  .  .  gene_id "."; transcript_id ".";
Chr1  mpileup indelregion  39405  39434  .  .  .  gene_id "."; transcript_id ".";
```

This file specifies the genomic range it will mask based on indel detection and the INWIN. Not all ranges are the same size because often there are multiple indels within a region.

The *gtffile* file is then uses to remove the blacklisted indel regions and produce a final .*sync* file called *phenoAB.sync*. We can see how many SNPs were lost through the indel region masking process:

```
> grep "indel" example_log.out
```

```
ALERT: With an indel window of 15 bp you lost 1435 SNPs (11%)
```

11% may be a lot of data to throw away and we can certainly tune indel windows in future runs. The impact of indel regions on false-positive SNP-calling is expected to be small based on simulation and thus a small INWIN may be appropriate. However, the commonly-used *GATK* package does allow for local realignment around indel regions to retain all SNPs within these regions. *GATK* is not currently built into the pipeline, yet if local realignment is a huge concern, users can use *GATK's* SNP calling and indel realignment in place of *bcftools* SNP calling and *Popoolation2* indel region removal.

vii) Frequency creation

After the final *sync* file (*phenoAB.sync*) is created, the R script, *r_frequency.R*, is initiated to estimate allele frequencies at each position. There are two frequency tables created, *phenoAB_full.fz* and *phenoAB_complete.fz* . The full table includes frequencies at each genomic position, even when a single population has 0 depth of coverage (its allele frequency will be 'NA'). The complete table includes frequencies at genomic positions that have at least a depth of coverage at 1 across all population included in the analyses (thus removing NA positions). The format of these tables are as follows:

> head -3 phenoAB_full.fz

Chr	Pos	Ref	A1	1	2
Chr1	6722	G	A	0.2	1
Chr1	10101	C	T	0.5	0.5

The columns are chromosome (Chr), position (Pos), reference genome allele (Ref), reference empirical allele (A1), A1 frequency for pop_1 (1), and A1 frequency for pop_2 (2).

In row 1, pop_1 has an allele frequency of 20% A, 80% G, and pop_2 has an allele frequency of 100% A.

Note that A1 is not synonymous with an alternate allele, it is simply a reference allele. In cases where the Ref allele is the most common allele in all populations, Ref=A1. If you want to know what the alternate allele is, this information is stored in the VCF file.

r_frequency.R uses frequencies to perform multiple filters, which will show up in the OUTDIR/filters directory. First, it performs a second minor allele frequency check at the specified MAF. This can be slightly different than the MAF filter performed on the VCF because libraries of the same category are merged in the *sync* file, yet split up in the *VCF* file.

> *phenoAB_MAF_fail.txt*

Indicates that just one additional SNP failed MAF when populations were combined

Next, the R script marks sites where >2 alleles were called. Though biologically possible between populations of diploid organisms, in most cases this is a red flag for duplicated regions or paralogs. In the filters folder:

phenoAB_norm_poly_one.txt	indicates positions where at least one population had >2 alleles
phenoAB_norm_poly_half.txt	indicates positions where half of the populations had >2 alleles
phenoAB_norm_poly_all.txt	indicates positions where all populations had >2 alleles

These lists can be incorporated into a blacklist for *PPanalyze*.

Finally, the R script determines coverage for each population at each site in *phenoAB_coverage.txt*

viii) Individual analyses

With INDCONT=on, *PPalign* performs additional analyses and assumes that each *fastq* file that was provided belongs to a different individual. The main purpose is to understand each individual's contribution to each SNP, provide additional filters, and provide an alternative normalization step.

It is worth noting that R scripts in these analyses can use a lot of memory. It is advised to not perform these analyses where the number of individuals exceeds 60 for any population and the number of filtered SNPs exceeds 20 million.

INDCONT creates large temporary files to count allelic contribution for each individual. In the inds folder:

> *_ind_stats.txt* indicates SNP contribution of each individual. Individuals are in the same order of how they are displayed in OUTDIR/pops

```
> head -3 pop_2_ind_stats.txt
```

##Mean.cov	Mean.non.zero	Max.cov	std.non.zero	n.pos	proportion.SNPs
1.797	2.133	9	1.201	11720	0.842
1.736	2.072	10	1.144	11660	0.838
1.547	1.939	9	1.101	11102	0.798

OUTDIR/pops/*pop_2.txt* indicates that the order is P201, P202, and P203.

The columns specify the mean coverage (Mean.cov), mean non-zero coverage (Mean.non.zero), maximum coverage (Max.cov), non-zero standard deviation of coverage (std.non.zero), number of SNPs represented (n.pos), and the proportion of SNPs represented of each individual (proportion.SNPs).

In this example, each individual looks relatively uniform. On average they contribute similar mean coverage to each SNP they represent (~2 X), and cover a similar proportion of SNPs (~ 80%). If not uniform, there may be less confidence in allele frequency estimates and the *sync* normalization step may be preferred.

> *_snp_stats.txt* indicates stats for each SNP.

```
> head -4 pop_1_snp_stats.txt
```

##CHR	pos	Mean.coverage	Sum.coverage	Number.individuals	Maximum.ind	Maximum.%.ind	Ind.%	Std
Chr1	19	1.5	3	2	2	0.667	1	0.707
Chr1	6722	2.5	5	2	4	0.8	1	2.121
Chr1	10101	1	2	1	2	1	0.5	NA

For each genomic position, we have the mean individual coverage (Mean.coverage), total coverage (Sum.coverage), number of individuals represented at that site (Number.individuals), maximum coverage contribution by a single individual (Maximum.ind), the proportion of the maximum individual contribution (Maximum.%.ind), percent of individuals represented at that site (Ind.%), and coverage standard deviation between represented individuals (Std).

_snp_stats.txt files can be rather large and their main purpose is to be used to blacklist additional SNPs for downstream analyses. For instance, let's say we want to remove sites that are only represented by a single individual:

```
> awk '$5 < 2' pop_1_snp_stats.txt | awk '{print $1,$2}' > black_list.txt
```

This command filters the Number.individuals column ($5) by individuals less than 2, then pipes to only print the first 2 columns (Chr,Pos) and writes this out to a file called *black_list.txt.*

Or, perhaps we want to mark SNPs with a standard deviation that is greater than or equal to 2:

```
> awk '$9 >= 2 || $9 ==NA' pop_1_snp_stats.txt | awk '{print $1,$2}' > black_list.txt
```

Note that I specify that stdev is >= 2 OR (||) ==NA. NAs occur when no st dev is present (*i.e.*, 1 individual)

We don't have to write these SNPs to a file either, we can also check how many SNPs fail our certain criteria:

```
> awk '$9 >= 2 || $9 ==NA' pop_1_snp_stats.txt | wc -l
```

Indicates 8939 SNPs would be removed if these criteria are used.

Use the *_snp_stats.txt* files to build unique black lists based on what makes sense with your data and organism.

Finally, INCONT performs a normalization step that produces a unique *.sync* file. This step normalizes contribution of each individual.

Using a hypothetical example, say there are 4 individuals represented at a SNP:

IND1) 20 T 40 A
IND2) 5 T, 2 A
IND3) 30 T 1 A
IND4) 0 T 3 A

The method essentially corrects these individuals as such:

IND1) 1 T 1 A
IND2) 1 T 1 A
IND3) 2 T 0 A
IND4) 0 T 2 A

IND1 we are pretty is heterozygous, so it contributes one of each of its alleles.
IND2, though with low coverage, also appears to be heterozygous and now holds equal weight to IND1.
IND3 is technically sequenced at 2 alleles, but the analysis will detect the large imbalance in alleles, and assume homozgygosity (this is adjustable); thus, IND3 contributes a homozygous genotype.
IND4 appears homozygous and contributes a homozygous genotype.

In this example the normalization essentially changes:
46 A (45%) 55 T (55%) to 4 A (50%) to 4 T (50%)

Other scenarios: If an individual only contributes a single allele (i.e., 1X of an allele), it will continue to only contribute 1 allele in this method. If an individual has >2 alleles at a site, its contribution will be discarded for that SNP.

In the end, this is a pseudo-genotyping method that attempts to reduce noise that may occur due to over-representation of a few individuals.

In the main OUTDIR folder, compare *phenoAB_norm.sync* (normalized sync file) to *phenoAB.sync* (standard synce file). There are also frequency tables produced for the normalized sync file (*i.e.*, _norm_full.fz) where you can determine how allele frequencies have changed.

3) PPstats

a) Prerequisites

PPstats is an optional module that produces some stats and plots to assess sequencing and alignment performance. It only requires *phenoAB_stats.mpileup* created by *PPalign* and the genome index* (*PP_genome.fa.fai*) to generate stats. Create another symbolic link if you want to run it directly from the command line:

```
> $ ln -s /usr/local/bin/poolparty/PPstats.sh  /usr/local/bin/PPstats
```

*Note that naming convention of chromosomes and scaffolds currently has a large impact on the plotting of stat results. This is because the R plotting script attempts to split chromosomes and scaffolds and order them numerically. The naming of chromosomes in some NCBI assembly genomes, which contain periods, follows a format that is not yet supported. Ideally chromosomes and scaffolds would follow a format of Chr01, Chr02, Chr03, *etc.* whereby chromosomes are easily identifiable and sorted by their order in the genome.

b) Editing the config file.

> *pp_stats.config*

#Specify Files#
FAI=/usr/local/bin/poolparty/example/PP_genome.fa.fai
The indexed genome file we previously generated, with path.
MPILEUP=/userdir/phenoAB/phenoAB_stats.mpileup
The large _stats.mpileup file we previously generated, with path.
OUTDIR=/userdir/phenoAB/stats
Choose a unique output directory for stats file
OUTFILE=PP_stats.txt
Choose a unique name for the stats output file

#Parameters#
SCAFP=Scaff
If there are both anchored chromosomes and unanchored scaffolds in the reference assembly, *PPstats* will attempt to differentiate the two. *PPstats* looks for scaffolds beginning with SCAFP and treats those as unanchored portions of the genome. In the example genome, scaffolds are denoted by "Scaff."
THREADZ=5
PPstats will use multiple processors with low memory usage. Choose maximum number of parallel-processes to run at once.
MINCOV=2
Minimum depth of coverage to retain a genomic position. Primarily used to determine the breadth of the genome covered by high depth of coverage reads. Set very low for this example to MINCOV=2.
MAXCOV=250
Maximum depth of coverage to retain a genomic position. Set to MAXCOV=250 for this example.

c) Running PPstats

Switch directory to where the config file and run:

```
> PPstats pp_stats.config
```

PPstats will provide rough status updates for the 8 blocks it runs through. The example should finish in less than a minute.

d) Output files

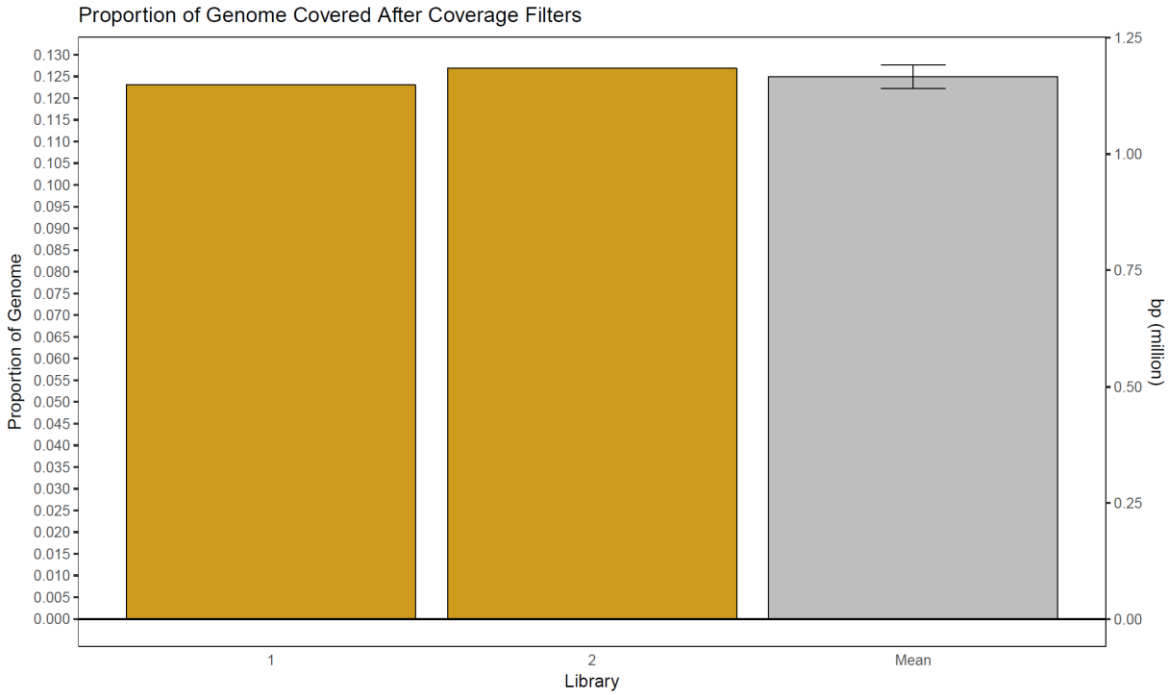
In the specified output folder, there will be a number of *txt* files and *pdfs*

The specified output file (*PP_stats.txt*) contains the raw statistics generated by *PPstats*. The first column of this file contains a unique string that is an identifier for the different analyses. The R script parses the file by the unique identifiers to make additional tables and plots. The string meanings are:

SUMMARY = summary information about the reference genome and number of populations
SCAFFOLD = information on scaffold length
COMB_TOT_BP = Total base pairs with sufficient read depth across all populations
COMB_TOT_PROP= Proportion of ref genome covered by reads with sufficient coverage
TMA = Total mean coverage (X) before filters of population
TMS = Standard deviation of coverage before filters of population
FMA = Filtered mean coverage (MINCOV > N < MAXCOV) of population
FMS = Standard deviation of mean coverage (MINCOV > N < MAXCOV) of population
FBC = Number of bp of ref assembly covered after filters (MINCOV > N < MAXCOV)
FPC = Proportion of ref assembly covered after filters (MINCOV > N < MAXCOV)
SCAF = Scaffold bp and proportions covered
ANC = Anchored bp and proportions covered
DOC/P = Proportion/bp of ref assembly covered at different minimum depths of coverage
CHRC/P = Proportion/bp of chromosomes covered after coverage filters

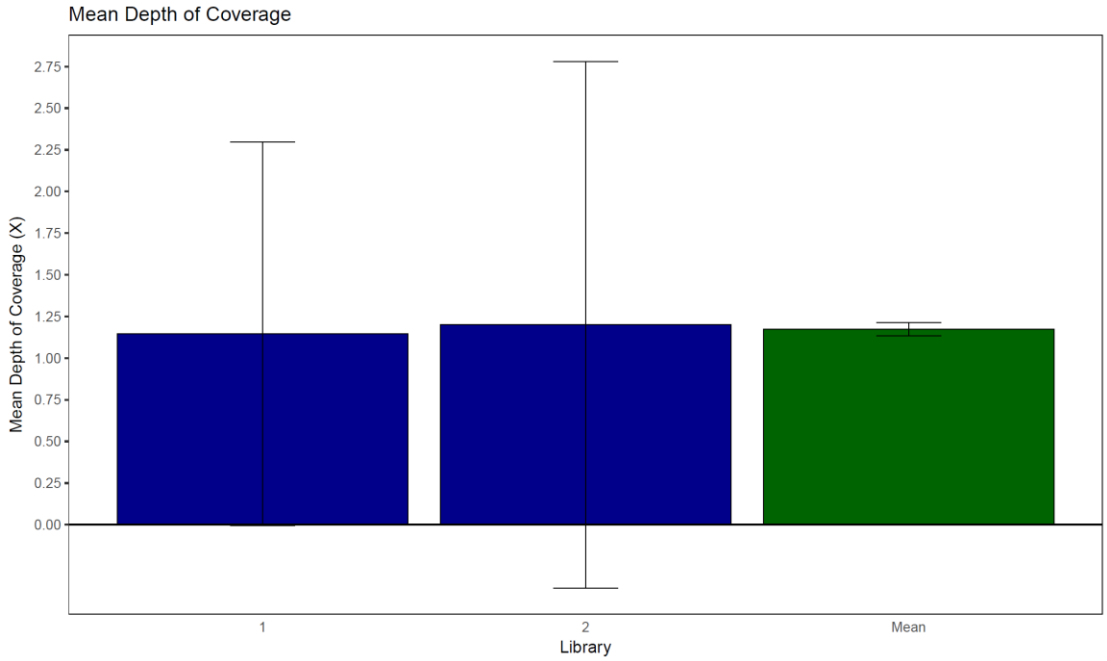
If run successfully, R will produce output plots and corresponding tables:

> *PP_stats_prop_cov.pdf*



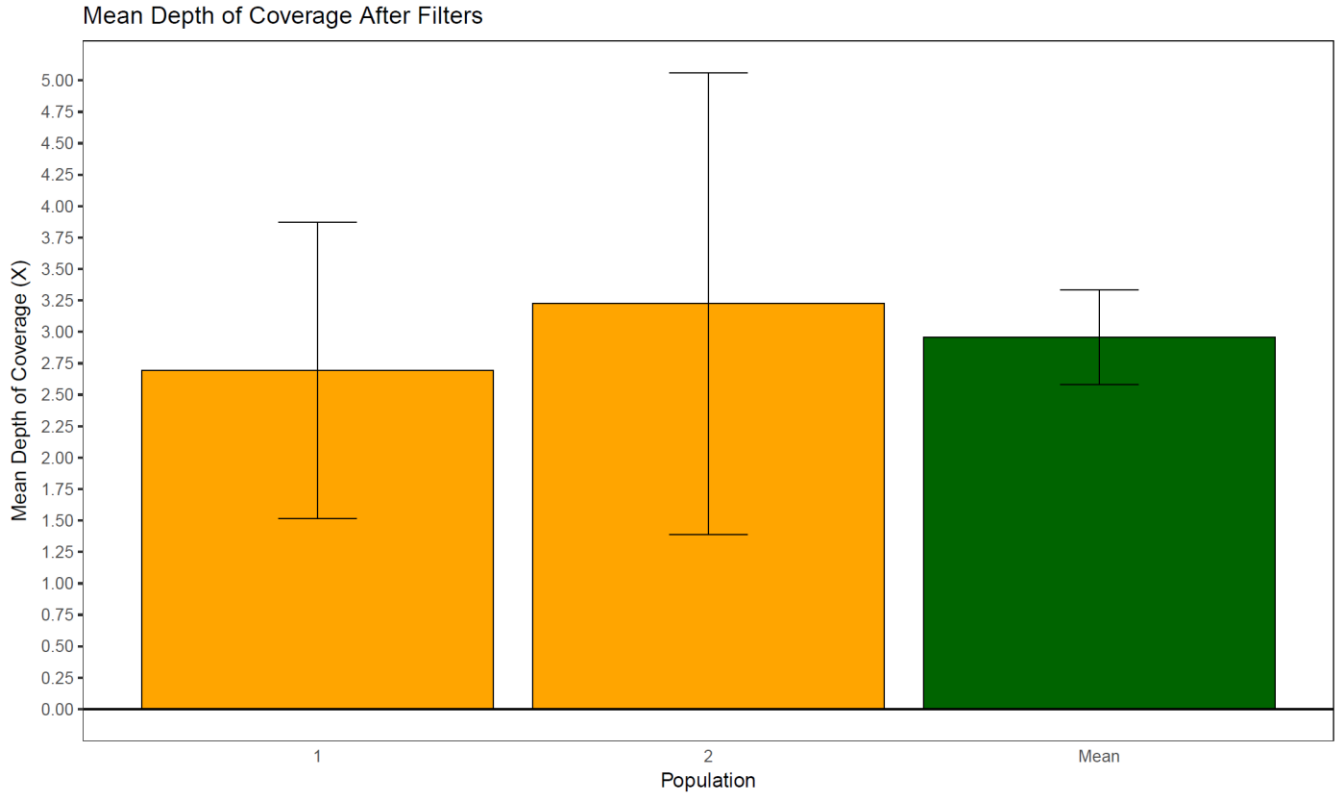
After coverage filters, the example libraries cover about 12.5% of the reference genome. Mean shows the mean of the two libraries and standard deviation between the two.

> *PP_stats_mean_coverage.pdf*



Mean depth of coverage before coverage filters showing extremely low average coverage. Bars represent standard deviation for each library (1,2) and standard deviation between libraries (mean)

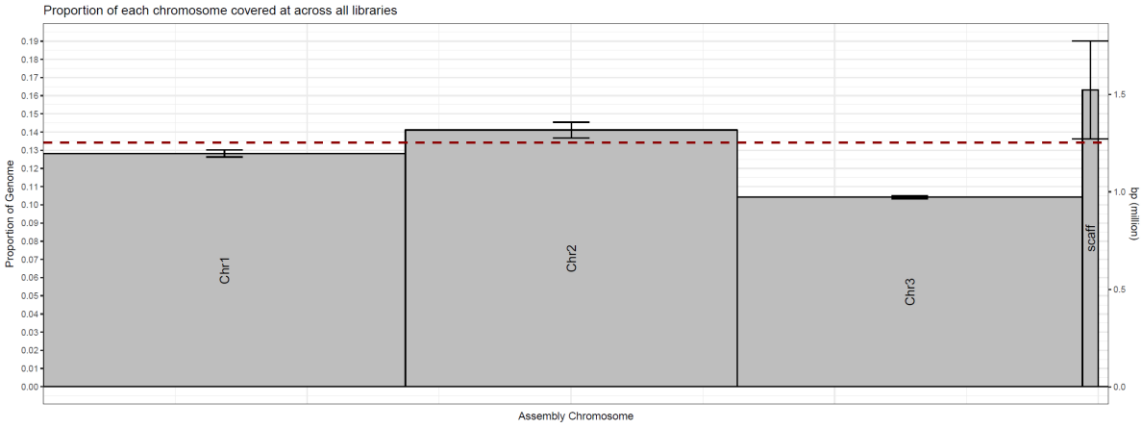
> *PP_stats_mean_filt_cov.pdf*



The mean coverage improves a bit when setting MINCOV=2

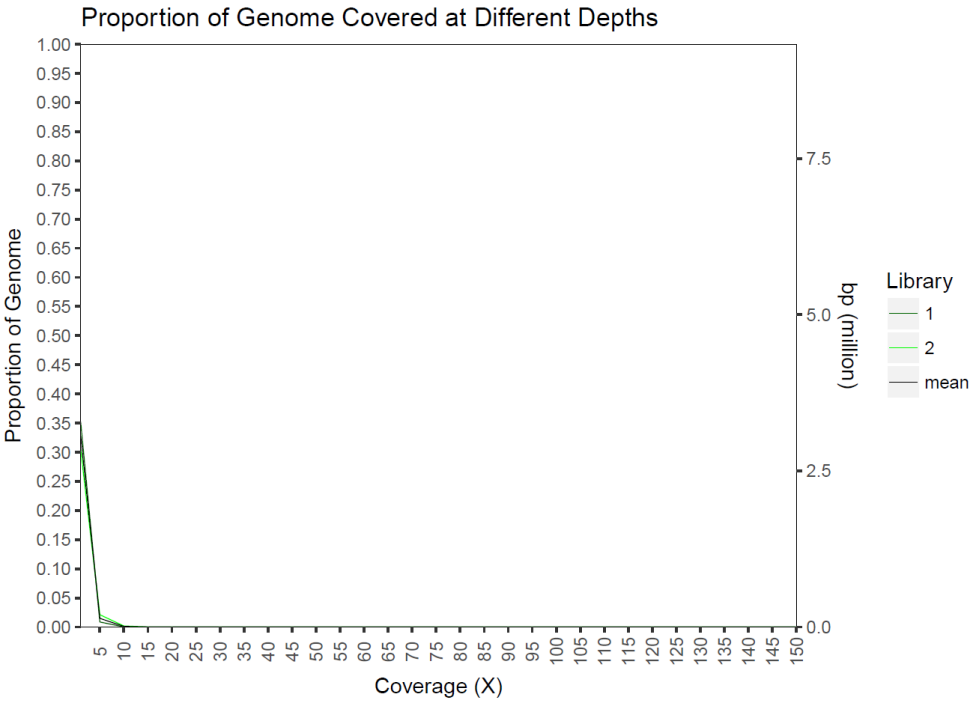
> *PP_stats_chr_prop_mean.pdf*

We can look at other plots that show chromosome alignment uniformity for each population, or anchored vs. unanchored plots, but this plot shows average uniformity across chromosomes and scaffolds. If there are too many chromosomes (>100) this plot will be skipped.



On average, after the MINCOV and MAXCOV filters, we can see that Chr3 is underrepresented and the scaffold is over-represented (based on the dotted-red average line). The widths of bar correspond to the size of each chromosome / combined scaffolds

> *PP_stats_prop_at_covs.pdf*



Though not impressive in our example, this plot shows how much of the genome is covered at different depths of coverage. It helps determine coverage threshold selection and indicates how much data may be lost if minimum thresholds are increased. In our example, if we increased the minimum threshold to 5X we would only be covering about 2.5 % of the reference assembly.

In general, these plots are useful for determining how much of the reference assembly your reads are covering and any potential biased alignment that could occur due to various library preparation protocols.

4) PPanalyze

PPanalyze is the final module of *PoolParty* and performs comparative analyses with the primary intention of discovering differentiated genomic regions between specified comparisons. *PPanalyze* produces output formats that can quickly be sorted or plotted with various tools.

Again, we can create a symbolic link to run *PPanalyze* from the command line

```
> $ ln -s /usr/local/bin/poolparty/PPanalyze.sh  /usr/local/bin/PPanalyze
```

a) Editing the config file.

The config file uses some of the previously generated files from *PPalign* to properly filter and analyze user-specified comparisons.

> pp_analyze.config

#Populations for analysis#

POPS=1,2

Specify the populations that will be compared, separated by a comma. In our example, we only have two categories so there is only one comparison that can be made. In other cases, you may have a sync file prepared by *PPalign* that has > 2 populations, and perhaps you want to look at differentiation between certain populations in this file.

If we hypothetically had 5 aligned populations, we can look at the average differentiation between all populations with POPS=1,2,3,4,5

Or perhaps we only want to know what is going on between population 1 and 5: POPS=1,5

Another possibility is that certain populations share a similar phenotype or characteristic, and should not be compared to one another. Let's say populations 1-3 has phenotype "X" and populations 4-5 have phenotype "Z". We can look at differentiation between "X" and "Z" with the specification of: POPS=1:2:3,4:5

This tells *PPanalyze* to average all differentiation between 1,2,3 and 4,5 but not incorporate differentiation within these groups. If you have sampling like this, it is also highly recommended to implement Popoolation2's CMH test (explained more below).

#Input files and names#

PREFIX=phenoAB_analyze

A unique prefix that will be given to all files produced.

COVFILE=/userdir/phenoAB/filters/phenoAB_coverage.txt

The coverage file that was produced by *PPalign*, with path. This will be in the filters directory and is used to filter out SNPs by comparison-specific coverage.

SYNC=/userdir/phenoAB/phenoAB.sync

One of the sync files generated by *PPalign*, with path. If this was an individual analysis, you have the option of selecting the normalized sync file as well. Use *phenoAB.sync* for this example.

FZFILE=/userdir/phenoAB/phenoAB_full.fz

Frequency file generated by *PPalign*, with path. The full table is recommended, you may also choose the normalized frequency file if the individual normalization analysis was performed.

BLACKLIST=/OUTIDR/ filters/*phenoAB_poly_all.txt*

Optional blacklist, with path, that is simply one genomic position per line (CHR POS) of loci that you wish to remove. In this case we are just using sites that have >2 alleles in all populations as a blacklist; however, this list can encompass a combination of any SNPs you wish. For instance, if you found SNPs that are dominated by single individuals, or know of structural variant regions in the ref assembly that you wish to avoid, you can include those SNPs in this list.

OUTDIR=/userdir/phenoAB/filters/phenoAB_poly_all.txt

Output directory for files. Note that a temporary directory will be created in this directory as well, but will be deleted upon completion of the module

#Types of Analyses#

FST=on

Single-SNP FST analysis on filtered SNPs. There are two supported types of FST analysis (see below)

SLIDINGFST=on

Sliding-window FST analysis on filtered SNPs. Similar to the FST above, except averages a user-specified window.

FET=on

Fisher's exact test for differences in allele frequencies. Produces P-values for each SNP and can be a useful output for performing other windowed analyses such as Local Score. Note, to perform the FET analysis you must have a 2-tailed *perl* module installed. To install this, simply run cpan Text::NSP::Measures::2D::Fisher::twotailed in the command line with admin privileges.

NJTREE=on

Calculates SNP density at a user-specified window and created a neighbor-joining tree if >2 populations are present in the analysis.

#Global Parameters#

MINCOV=2

Minimum coverage needed to retain a SNP. MINCOV=2 means that both pop_1 and pop_2 must have a minimum coverage > 2. The more populations that are included in an analysis, the more SNPs will be lost due to minimum coverage not being met across all populations. Set to a threshold of 2 for this example.

MAXCOV=100

Maximum coverage needed to retain a SNP. Serves generally as a paralog filter, yet the value should depend on the organism and coverage from *PPstats*. For instance, it makes sense to base this value on the mean coverage of all reads, and set it to a set number of standard deviations above the mean. If paralogs or duplication is not an issue in your organism, this value can be set very high. For this example, with very low mean coverage, leave at 100.

MAF=0.06

Analysis-specific minor allele frequency. Though global MAF filters have already been applied, this value is based solely on user-specified comparisons. In our example, we have already filtered out MAF < 0.05 between pop_1 and pop_2, so let's set MAF to a high value to filter SNPs further.

#FST Parameters#

FSTTYPE=traditional

Type of FST analysis to perform for both FST and SLIDINGFST. The two options are traditional or an equation by karlsson (Karlsson *et al.* 2007). The karlsson method considers allele counts which makes it appropriate for pooled data. Stick with traditional for the example.

WINDOW=10000

Window size in bp for FST window analysis. This will be the window size, according to the ref genome, in which FST is averaged.

STEP=5000

Step size for SLIDINGFST. Must be smaller than window size.

NIND=5

Number of individual for FST analyses. Used for sample size correction.

#NJ tree Parameters#
STRWINDOW=10000

The window size to look for SNP density and in which allele frequencies are combined. If set to 1, will not calculate SNP density and will not use a combination method for allele frequencies.

BSTRAP=2000

Number of bootstraps to perform on neighbor-joining (NJ) tree. The NJ tree uses Nei's genetic distance and will produce a consensus tree with node support as well as a single tree with node support. The NJ tree will not be produced for the example since there are less than three populations.

AFFILT=1

Allele frequency filter for NJ trees. This is a crude method to attempt to remove adaptive regions of the genome that may overpower neutral structure. In general, should be set to 1, which means do not filter allele frequencies, or filter allele frequencies greater than 1 (which is impossible).

METHOD="mean"

Method to use when investigating SNPs within a window size for the NJ tree. If SNP density is not uniform due to biased alignment, then certain regions of the genome may contribute more to identifying population structure. The windowed approach in NJ tree attempts to dampen these effects by looking at allele frequencies of SNPs within a window and either taking the mean ("mean"), choosing the SNPs with least deviation in the window ("sdmin"), the most deviation in the window ("sdmax"), a random SNP in the window ("random"), the SNP with the largest allele frequency difference ("rangemax"), the SNP with the smallest allele frequency difference in ("rangemin"), or the first SNP of the window ("first").

#Dependencies#
POOL2=/usr/local/bin/popoolation2_1201/

The *Popoolation2* directory is the only dependency for this package.

b) Running PPanalyze

Run *PPanalyze* similar to other modules:

```
> PPanalyze pp_analyze.config
```

The run should take < 5 min to complete. Similar to PPalign, alerts are printed that indicate the number of SNPs removed from filters.

An important ALERT here is "ALERT: Pops 1 2 are now in the order of 1 2 in the subset sync file." For the example, the order of populations has not changed; however, if a subset of populations is specified, and done so in a different order, the resulting sync file will be in that order. The aforementioned line thus indicates which order the populations are in after sub-setting.

c) Output files

Within the output files are raw Popoolation2 outputs ("_raw"), and outputs formatted for plotting ("_analysis").

```
> awk '$4 > .75' phenoAB_analyze_analysis_fst.txt | wc -l
```

We see that 137 SNPs had very high FST (>0.75)

```
> awk '$4 < .05' phenoAB_analyze_analysis_fet.txt | wc -l
```

Only 83 SNPs had significant allele frequency differences (p <0.05), likely due to the low coverage allowed in the example.

Additionally are analysis-specific *.sync* files and frequency tables if additional analyses are desired. For instance, *Popoolation2* can perform a Cochran–Mantel–Haenszel (CMH) test on subset sync files to look for consistent changes in allele frequencies between biological replicates. If you were to have sampling from different locations: phenotype A from location 1, phenotype B from location 1, phenotype A from location 2, and phenotype B from location 2, you can identify consistent deviations in allele frequencies between phenotype A and B from different geographic localities. Popoolation2 documentation explains how to run the test.

d) Plotting results

There are numerous ways to visualize results of differentiation. Provided here is an example using *qqman* in R (Turner 2014). Other python-based (matplotlib) or perl-based (e.g., Circos; Krzywinski *et al.* 2009) packages would work as well.

Plotting anchored portion of the sliding window FST results:

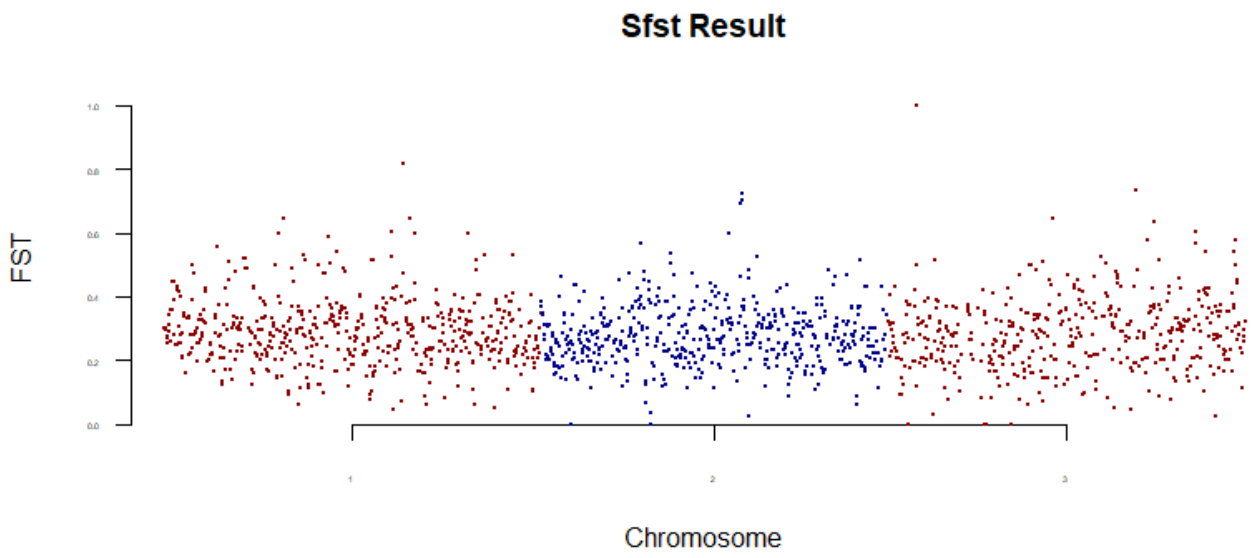
In R (RStudio):

```
library(qqman)
# read in the sfst results
snps <- read.delim("phenoAB_analyze_analysis_Sfst.txt",  header=FALSE, sep = " ")
#Apply qqman friendly header
colnames(snps) <- c("CHR", "BP", "SNP", "FST")

# We are only interested in anchored chromosomes in this example, remove Scaffolds
snps <- snps [!grepl("Scaff", snps $CHR),]

#for qqman , chromosome names must be an integer
snps$CHR <- as.integer(gsub('[a-zA-Z]', '', snps$CHR ))

#run manhattan, for options see documentation on qqman
manhattan(snps, suggestiveline = F, ylim=c(0.00, 1.05),
                    p="FST", logp=FALSE, col = c("darkred", "darkblue"),
                            =cex=0.1, cex.axis = 0.3, ylab= "FST", xlab= "Chromosome",
                            main= "Sfst Result")
```



While average FST is high, if this were a real biological example, we would want to further investigate some of the regions with very high FST, as they might contribute to the difference between the "X" and "Z" phenotype.

5) Beyond PoolParty

Depending on the question and genomic resources available, there are many branching paths of analysis after *PoolParty*. For instance, if gene annotation is available, it may be useful to identify which genes highly differentiated SNPs represent. Enrichment of GO categories is also a common practice. Even if annotations are not available, running BLAST on sequences around genes of interest may give insight to the genomic basis of traits of interest.

Some *PoolParty* outputs, such as FET tests, can be used for more intricate statistical analyses of differentiation that control for noise or false-positives. Local score (Fariello *et al.* 2017) and GLMS (Wiberg *et al*. 2017) are two additional methods that seek to achieve this. Depending on the structure of libraries, outlier tests that control for population structure and use allele frequency data, such as FLK (Bonhomme *et al.* 2010), may help differentiate adaptive from neutral variation.

6) References

Bonhomme, M., Chevalet, C., Servin, B., Boitard, S., Abdallah, J., Blott, S., & SanCristobal, M. (2010). Detecting selection in population trees: the Lewontin and Krakauer test extended. *Genetics*, 186(1), 241-262.

Fariello, M. I., Boitard, S., Mercier, S., Robelin, D., Faraut, T., Arnould, C., ... & Gourichon, D. (2017). Accounting for Linkage Disequilibrium in genome scans for selection without individual genotypes: the local score approach. *Molecular Ecology*.

Karlsson, E. K., Baranowska, I., Wade, C. M., Hillbertz, N. H. S., Zody, M. C., Anderson, N., ... & Comstock, K. E. (2007). Efficient mapping of mendelian traits in dogs through genome-wide association. Nature genetics, 39(11), 1321.

Krzywinski, M., Schein, J., Birol, I., Connors, J., Gascoyne, R., Horsman, D., ... & Marra, M. A. (2009). Circos: an information aesthetic for comparative genomics. *Genome research*, *19*(9), 1639-1645.

Turner, S.D. qqman: an R package for visualizing GWAS results using Q-Q and manhattan plots. biorXiv DOI: 10.1101/005165 (2014).

Wiberg, R. A. W., Gaggiotti, O. E., Morrissey, M. B., & Ritchie, M. G. (2017). Identifying consistent allele frequency differences in studies of stratified populations. *Methods in Ecology and Evolution*.