1000 genomes phase 3 low coverage WGS dataset

The low coverage WGS dataset used in the present study represents 2535 samples from 26 geographically distinct populations sequenced to a minimum 3X coverage as part of the 1000 genomes project phase 3 release. This set of 2535 samples includes 2504 unrelated individuals of the final phase 3 panel plus 31 individuals related to the main panel. For a list of the 31 related samples see https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/20140625_related_individuals.txt.

Sequence alignment pipeline used by 1000 genomes project consortium

The high coverage dataset sequences were produced using Illumina HiSeq 2000 platform with 70bp reads. Reads were aligned to Integrated human reference sequence hs37d5 consisting of GRCh37 reference genome build with decoy sequences. For reference genome details see:
https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/phase2_reference_assembly_sequence/README_human_reference_20110707
The initial run-level sequence alignments were produced using bwa v0.5.9 (*bwa aln -q 15*) generating SAM files. Samtools were then used to create sorted BAM files (*view, sort*), fix mate-pair information (*fixmate*) and add MD tags (*fillmd*). GATK was used to perform further local realignment around known Indels (*IndelRealigner*) and recalibrate the realigned BAM read qualities (*CountCovariates, TableRecalibration*), with NM tags fixed and BQ tags added with Samtools (*calmd*). Then, Picard was used to merge the run-level BAMs (*MergeSamFiles*) and mark duplicates (*MarkDuplicates*), producing the final sample-level alignment BAM files. The final quality control was applied to the sample-level BAMs, checking for sufficient coverage, not excessive number of short indels, and no evidence of sample contamination. The final set of 2533 QC-passing alignment files, made available on 1000genomes.ebi.ac.uk ftp server at https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/, are the source of the low coverage WGS mitochondrial sequence data we use in the present study.

For detailed description of the alignment pipeline see:
https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/README.alignment_data

Sequence alignment index file

1000 genomes project phase 3 panel low coverage WGS dataset:
https://ftp.1000genomes.ebi.ac.uk/vol1/ftp//phase3/20130502.phase3.low_coverage.alignment.index

MT sequence data retrieval and initial processing: Samtools

The low coverage mitochondrial sequence dataset used in the present study was obtained by extracting reads mapped to the MT genome from the 2535 low coverage whole genome sequence alignment BAM files made available by the 1000 genomes project consortium. SAMtools was used to extract the reads mapped to the mitochondrial genome and generate pileup format alignment files for each sample.

Pipeline

STEP 1: Retrieve reads mapped to MT genome and save locally in BAM format

```
samtools view -b -X path/to/remote/BAM path/to/remote/BAI MT:1-16569 -o output.bam
```

Inputs

`path/to/remote/BAM`	Full alignment in BAM format
`path/to/remote/BAI`	The corresponding alignment index file in BAI format

Outputs

`output.bam`	Extracted mitochondrial alignment in BAM format

Options

`MT:1-16569`	Set region of interest to full MT genome
`-b`	Specify output in BAM format
`-X`	Specify alignment index file location

STEP 2: Index the created BAM file

```
samtools index -b input.bam
```

Inputs

`input.bam`	Mitochondrial read alignment in BAM format

Outputs

`input.bam.bai`	Index file saved in the same directory as the input BAM file

STEP 3: Generate pileup format TXT file

```
samtools mpileup -f rCRS.fa -d0 -a input.bam -o output.txt;
```

Inputs

`rCRS.fa`	rCRS reference sequence in FASTA format
`input.bam`	Mitochondrial read alignment in BAM format

Outputs

`output.txt`	Text file in pileup format

Settings

`-q25`	Set minimum read mapping quality (MAPQ): 25
`-d0`	Remove limit on maximum number reads to include
`-a`	Output all positions, including those with zero depth
`q/Q not specified`	Use default base quality & mapping quality settings

Low coverage dataset mitochondrial consensus sequences

Sample mitochondrial genome consensus sequences were produced and made available by the 1000 genomes project consortium as part of the project phase 3 release, available on the 1000 genomes project ftp server at ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/MT/chrMT_sequences_2534.20160101.fasta.gz. In our study, we have used these publicly available MT consensus sequences in our analyses as the basis for distinguishing between reference and non-reference reads. File `1KG_No_Het.fasta` containing 2534 samples consensus sequences in FASTA format was retrieved from the ftp server, which we used for the later steps of our analyses.

This set of MT consensus sequences made available by the 1000 genomes project is however incomplete, with the consensus sequence for sample NA19159 missing from the dataset. This is a known issue with the phase 3 data release, which we circumvented by calling sample NA19159 consensus sequence locally with Samtools as follows:

```
samtools consensus -m simple -aq -r MT:1-16569 --show-ins no --show-del yes
-f fasta input.bam -o consensus.fa
```

Inputs

`input.bam`	Sample sequence alignment in BAM format

Outputs

`consensus.fa`	Sample consensus sequence in FASTA format

Options

`-m simple`	Use simple consensus calling mode (frequency counting algorithm)
`-r chrM:1-16569`	Include all MT genome positions
`-a`	Output all bases in region
`-q`	Use base quality scores
`--show-del yes`	Include deletions as `'*'`
`--show-ins no`	Exclude insertions
`-f fasta`	Specify output in FASTA format

Merged consensus sequence FASTA file '`G1K_phase3_MT_consensus_sequences.fa`' was produced by combining the 2534 sequences from '1KG_No_Het.fasta' with the locally called sample NA19159 consensus sequence, and was used in the subsequent steps for reference and non-reference read identification.

Low coverage MT sequence pileup parsing: Python

Pileup format

Pileup is a tab delimited text format where each line represents one sequence position and consists of 5 columns: (1) sequence identifier (str), (2) position number (int), (3) reference base (char), (4) number of reads covering the position (int), (5) bases called at this position (str) and, optionally, (6) base quality scores at this position (str).

The encoding used in the base call string:

Forward direction	**Reverse direction**	**Interpretation**
.	,	Reference match
ACTGN	actgn	Reference mismatch
^		Start of a read marker
Any single character following '^'		Read mapping quality as ASCII value -33
$		End of a read marker
*		Placeholder for 2nd deleted base onwards in a multi-bp deletion
+[length][sequence_inserted]		Insertion after this base
-[length][sequence_deleted]		Deletion after this base

STEP 1: Parse each sample pileup file to obtain a tally of all possible base calls found at any given MT genome position in that sample.

Script: `Pileup_parsing.py`

Inputs

`in_file.txt`	Sample-level sequence alignment file in pileup format
`ref_file.txt`	Text file with MT reference sequence (rCRS)

Outputs

`out_file.txt`	Tab-delimited output .txt file with position data and parsed read counts organised in 21 columns (headers: "Chromosome", "Position", "Reference", "Reads", "Total_As", "Total_Cs", "Total_Ts", "Total_Gs", "Total_Ns", "For_As", "Rev_As", "For_Cs", "Rev_Cs", "For_Ts", "Rev_Ts", "For_Gs", "Rev_Gs", "For_Ns", "Rev_Ns", "Insertions", "Deletions"), with each line corresponding to one mtDNA position

Algorithm structure

- Load rCRS reference sequence from `ref_file.txt` as a string
- Create empty output file `out_file.txt` and write the first line of column headers
- Read the input pileup file `in_file.txt` line by line. For each line:
  - Split the line into the Chromosome ID, Position, Reference base, Read count and a Pileup string
  - Parse the pileup string one character at a time as follows
    - Forward reference match (`"."`) – Increment the relevant forward base counter
    - Forward reference mismatch (`'A'/'C'/'G'/'T'/'N'`)- Increment the corresponding forward counter
    - Reverse reference match (`","`) – Increment the relevant reverse base counter
    - Reverse reference mismatch (`'a'/'c'/'g'/'t'/'n'`) - Increment the corresponding reverse counter
    - Start of an indel (`"+"/"-"`) – increment the insertion (`"+"`) or deletion (`"-"`) counter and move pointer forward to skip the indel sequence
    - Base in a multi-bp deletion (`'*'`) – increment the counter for deletions
    - Start of read marker (`"$"`) – move pointer to skip the character
    - End of read marker (`"^"`) – move pointer to skip this and the following quality score character
  - Assemble tab delimited output string containing position number, reference base, total read count and the parsed counts of different base calls and indels and write it as a new line into the `out_file.txt`
- If the pileup file is missing data for any positions, the reference base is read from rCRS and a line with 0s for all counters written to the output file for that position

STEP 2: Aggregate base count data from individual samples into a population level data files such that each output data file contains an array of base call counts with one row per sample and one column per mtDNA position.

Script: `Merge_counts.py`

Inputs

`[SampleID]_basecounts.txt`	Multiple tab-delimited text files generated by `Pileup_parsing.py`, each listing founts of different types of base calls recorded at each mtDNA position in a specific sample

Outputs: 19 tab delimited `.txt` files

`Sample_IDs.txt`	List of the processed sample IDs

`For_As.txt`
`For_Cs.txt`
`For_Gs.txt`
`For_Ts.txt`
`For_Ns.txt`
	Counts of A/C,G,T or N calls in forward sequencing at each mtDNA position (columns) in each sample (rows)

`Rev_As.txt`
`Rev_Cs.txt`
`Rev_Gs.txt`
`Rev_Ts.txt`
`Rev_Ns.txt`
	Counts of A/C,G,T or N calls in reverse sequencing direction at each mtDNA position (columns) in each sample (rows)

`All_As.txt`
`All_Cs.txt`
`All_Gs.txt`
`All_Ts.txt`
`All_Ns.txt`
	Total counts of A/C,G,T or N calls across both sequencing directions at each mtDNA position (columns) in each sample (rows)

`Insertions.txt` `Deletions.txt`	Number of reads recording insertions or deletions at each mtDNA position (columns) in each sample (rows)
`Reads.txt`	Total analysed reads at each position (columns) in each sample (rows)

Algorithm structure

> Create output directory
> Create 19 empty output files for recording sample IDs (`Sample_IDs.txt`), total read counts (`Reads.txt`), indel counts (`Insertions.txt`, `Deletions.txt`), and different base call counts in forward direction (`For_As.txt`, `For_Cs.txt`, `For_Gs.txt`, `For_Ts.txt`, `For_Ns.txt`), reverse direction (`Rev_As.txt`, `Rev_Cs.txt`, `Rev_Gs.txt`, `Rev_Ts.txt`, `Rev_Ns.txt`) and across both directions (`All_As.txt`, `All_Cs.txt`, `All_Gs.txt`, `All_Ts.txt`, `All_Ns.txt`), and write header line to each output file file.
> Access the input directory and loop through sample-level base count data files, processing each one at a time:
>> Extract sample ID from file name
>> Read sample-level base count data file columns into separate arrays
>> Insert sample ID at the start of each column-derived base count array and write the array as tab-delimited string into a new line of the appropriate output file
>> Append the column-derived data arrays as a tab delimited strings to the corresponding output files as new lines.
>> Append sample ID to `Sample_IDs.txt` file

Low coverage MT sequence allele frequency calculation: MATLAB

Step 1: **Import read count data & excluded truncated samples**

Load read count matrices from .txt files such that rows = samples and columns = positions and save all produced variables into .mat files. In this step data associated with samples HG00102 and NA19031 is removed due to partially missing sequence alignment data for these samples. The remaining dataset consists of 2533 samples.

Code: SCRIPT_1_Basecount_import.m

Outputs			**Dimensions & data type**		
SampleIDs			2533 x 1	string	7 letter G1K project sample IDs
Individuals			2533 x 1	double	Numeric IDs (original processing order)
Positions			1 x 16569	double	mtDNA position numbers
Reads_TOTAL			2533 x 16569	double	Total reads at each position in each sample
Basecalls	Forward	A	2533 x 16569	double	Base counts at each position in each sample. Rows – Samples Columns – Positions
		C	2533 x 16569	double	
		G	2533 x 16569	double	
		T	2533 x 16569	double	
		N	2533 x 16569	double	
	Reverse	A	2533 x 16569	double	
		C	2533 x 16569	double	
		G	2533 x 16569	double	
		T	2533 x 16569	double	
		N	2533 x 16569	double	
	All	A	2533 x 16569	double	
		C	2533 x 16569	double	
		G	2533 x 16569	double	
		T	2533 x 16569	double	
		N	2533 x 16569	double	
Indels	Insertions		2533 x 16569	double	Indel counts at each position in each sample
	Deletions		2533 x 16569	double	

Base call and Indel variables grouped into higher level data structures

Step 2: **Import MT reference and sample consensus sequences**

Import from FASTA file rCRS reference sequence (*rCRS.fa*) and 2533 sample consensus sequences (*G1K_phase3_MT_consensus_sequences.fa*). Apply following modifications to consensus sequences:
(1) replace all deletions marked as '*' with '−';
(2) set position 3107 as '−';
(3) change all letters to be upper case;
(4) set any non- A/C/G/T/N/− characters as 'N'.
Sort the sample consensus sequences to be in the same order as SampleIDs from the previous step. In this step consensus sequences of the excluded samples HG00102 and NA19031 are removed.

Code: SCRIPT_2_consensus_seq_import.m

Outputs	**Dimensions & data type**		
ConsensusSeqs	2533 x 16569	char	Sample consensus sequences
rCRS	1 x 16569	char	rCRS sequence
Sample_overview	*2533 x 16569 table*		Fields: Sample number, ID, consensus sequence

Step 3: Calculate total, reference and non-reference read counts and base frequencies

Excluding indels, calculate total read counts and base frequencies. Identify reference and non-reference reads and calculate reference and non-reference read frequencies.

Code: SCRIPT_3_Base_frequency_calculation.m

Inputs			**Dimensions & data type**	
Basecalls	Forward	A,C,G,T	*3 x 4 variables:* 2533 x 16569 double	Counts of reads supporting each of the base types A, C, G, or T
	Reverse	A,C,G,T		
	All	A,C,G,T		
ConsensusSeqs			2533 x 16569 char	Sample consensus sequences

Outputs			**Dimensions & data type**	
Reads	Forward		*3 variables:* 2533 x 16569 double	Total A + C + G + T base calls
	Reverse			
	All			
Frequency	Forward	A,C,G,T	*3 x 4 variables:* 2533 x 16569 double	A, C, G, or T read fraction of the total A + C + G + T count
	Reverse	A,C,G,T		
	All	A,C,G,T		
Reads_Ref	Forward		*3 variables:* 2533 x 16569 double	Reads that support individuals' consensus sequence base
	Reverse			
	All			
Reads_NonRef	Forward		*3 variables:* 2533 x 16569 double	Sum reads that do not support individuals' consensus sequence base
	Reverse			
	All			
Frequency_Ref	Forward		*3 variables:* 2533 x 16569 double	Fraction of reads that support consensus sequence base
	Reverse			
	All			
Frequency_NonRef	Forward		*3 variables:* 2533 x 16569 double	Fraction of all reads that do not support consensus sequence base
	Reverse			
	All			

Algorithm structure

> Load A,C,G,T base call counts (Basecalls) and sample consensus sequences (ConsensusSeqs)
> Calculate A+C+G+T base call count totals in forward, reverse and across both sequencing directions (Reads).
> Calculate each base type relative fraction of total ACGT counts (Frequency) for each sequencing direction and across both directions together
> For each of A,C,G,T base types obtain index of the locations where sample consensus sequences contain this base type. Use the location index for each base type to extract the corresponding subset forward, reverse and total read counts and place into reference read arrays (Reads_Ref), while adding the reads from all other locations to the non-reference read arrays (Reads_NonRef)
> Calculate reference and non-reference read relative fractions of the total read counts for forward and reverse sequencing directions and across both directions (Frequency_Ref, Frequency_NonRef)
> Apply data filtration to Basecounts, Reads, Reads_Ref, Reads_NonRef, Frequency, Frequency_Ref, Frequency_NonRef arrays by replacing data with 'NaN' at:
  >> Position 3107
  >> Positions where total A+C+G+T read count total in forward or reverse sequencing direction is <500
  >> Positions where sample consensus sequences contain characters other than A,C,G or T
> Write filtered output variables to csv files and save as .mat files for use in subsequent analyses