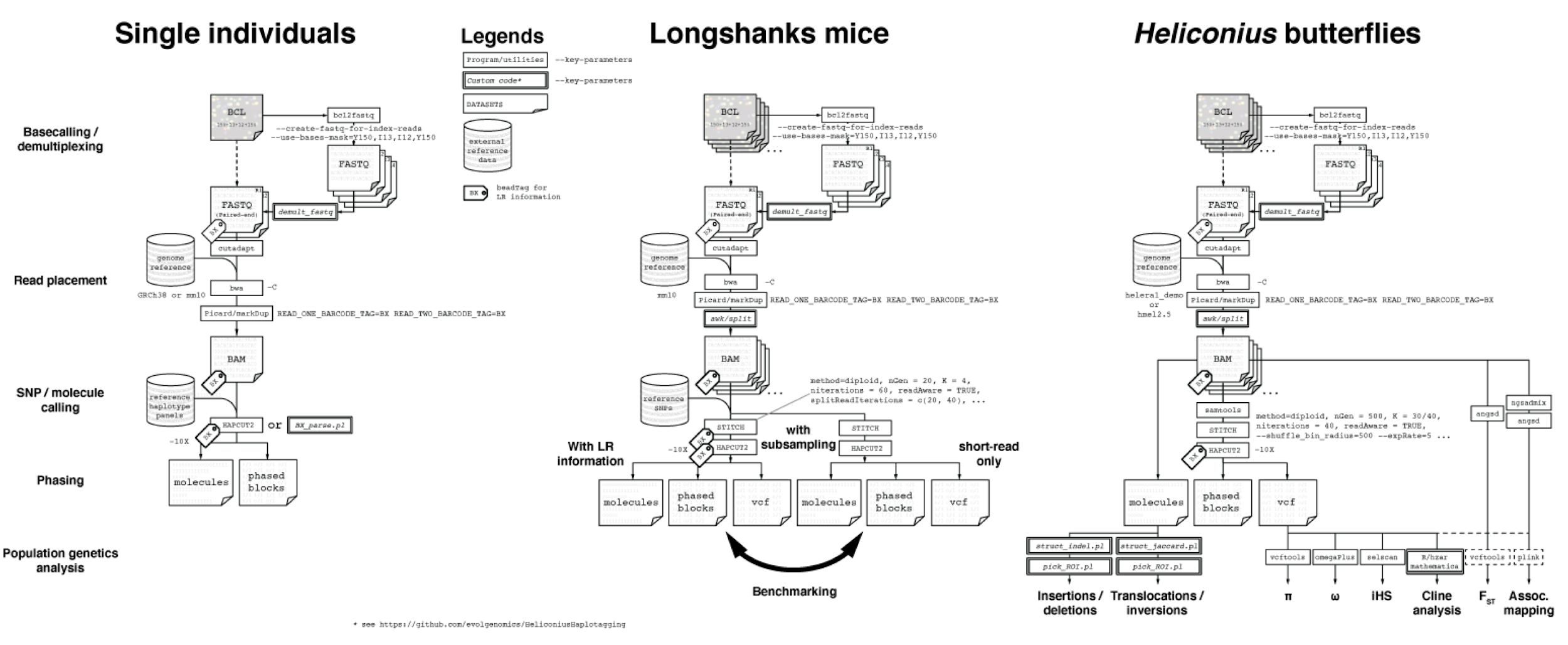
**Haplotagging analysis pipeline:**



1. **During sequencing:** add extra index cycles for the extended barcode information
2. **BCL to FASTQ**

Sequences arrive from Illumina machine in BCL (Binary Base Call) format. Typically sequencing facilities perform the next step - conversion to FASTQ using bcl2fastq:

Illumina sequencing instruments generate per-cycle BCL basecall files as primary sequencing output, but many downstream analysis applications use per-read FASTQ files as input. bcl2fastq combines these per-cycle BCL files from a run and translates them into FASTQ files. At the same time as converting, bcl2fastq also separates multiplexed samples (demultiplexing). The multiplexed sample FASTQ files are assigned to projects and samples based on a user-generated sample sheet, and stored in corresponding directories.

However, haplotagging requires custom settings for future read demultiplexing. At this stage barcodes are included in the header of the fastq as comment tags: BX, RX and QX.

BX: A/B/C/DXX human-readable beadTags

RX: raw reads themselves

QX: the quality string

Software used: bcl2fastq v2.17.1.14 (see Manual in KeyReferences folder)

Example of code:

bcl2fastq --use-bases-mask=Y150,I13,I12,Y150 --create-fastq-for-index-reads -r 20 -w 20 -d 20 -p 40 -R <run folder> \ --tiles s\_<REGEX> --output-dir=<outdir> --interop-dir=<Interop\_dir> --reports-dir=<Reports\_dir> --stats-dir=<Stats\_dir> 2> [STDERR.log] ./tag\_fastq\_13plus12.o <prefix> <OUTDIR/out\_prefix> 2>[log\_file]

1. Adapters trimmed with cutadapt, adapters sequences are in the example:

Adapter trimming:

cutadapt -a CTGTCTCTTATACACATCT -g AGATGTGTATAAGAGACAG -A CTGTCTCTTATACACATCT -G AGATGTGTATAAGAGACAG --cores=<cores> -O 5 -o <read\_1\_out.fastq.cutadapt.gz> -p <read\_1\_out.fastq.cutadapt.gz> --pair-filter both \ $dir/$fbname\\_R1\_001.fastq.gz $dir/$fbname\\_R2\_001.fastq.gz

Length filtering (>30 bp):

cutadapt -m 30 \ -o <read\_1\_out.fastq.cutadapt.gz> -p <read\_2\_out.fastq.cutadapt.gz> \ <read\_1\_out.fastq.cutadapt.1.gz> <read\_2\_out.fastq.cutadapt.1.gz> \ --too-short-output=<read\_1\_out.tooshort.fastq.gz> --too-short-paired-output=<read\_2\_out.tooshort.fastq.gz >

1. Mapping with BWA

Extra comment tags with barcodes will be used in downstream analysis, therefore -C flag is provided (otherwise mapping is standard):

bwa mem -C -t 50 <helera1\_demo\_dir>/Heliconius\_erato\_demophoon\_v1.fa \ $file ${file/R1\_001/R2\_001} \ -R "@RG\tID:$fbname\tSM:$fbname\tLB:$fbname\tPL:Illumina.HiSeq3000.2x150" | samtools view -bh - > /tmp/mkucka/$fbname.erato.bam

samtools sort \ -@ 50 -l 9 \ -T /tmp/mkucka/$fbname.tmpsort \ -o /tmp/mkucka/$fbname.erato.sorted.bam \ /tmp/mkucka/$fbname.erato.bam

1. Marking PCR and optical duplicate reads using MarkDuplicates.

In addition to standard settings, flags for reading BX tag are included: READ\_ONE\_BARCODE\_TAG=BX and READ\_TWO\_BARCODE\_TAG=BX (same parameter is specified for working with 10x)

java -Xmx12g -XX:ParallelGCThreads=64 -jar <picard\_dir>/picard.jar MarkDuplicates \ I=$file \ O=/tmp/mkucka/$fbname.pMarkdup.bam \ M=$dir/$fbname.pMarkdup.metrics \ CREATE\_INDEX=TRUE READ\_ONE\_BARCODE\_TAG=BX READ\_TWO\_BARCODE\_TAG=BX VALIDATION\_STRINGENCY=LENIENT

1. Molecule identification.

Molecules sharing the same beadTag (BX tag) are determined by using custom scripts (see Seafile project folder). Maximum gap size of 50 kbp is allowed. Basic statistics on the molecule size, molecular coverage and reads per molecule are provided as output.

## makePhaseSet.sh

Using awk creates PhaseSet file for the next step

1. BX\_parse.pl

Generates a BED file for each molecule from a BX-sorted BAM file, defined as reads that share the same BX tag with a maximum internal gap of 50kb.

#Usage:

#perl BX\_parse.pl <POS\_SORTED\_BAM> <PHASESET> <OUTPUT>

#

#PhaseSet is in a modified BED format, with the following columns: CHROM POS-1 POS HAP0 HAP1 COVERAGE

#Chr8 112 113 G A 15

#Chr8 118 119 A T 13

#Chr8 125 126 C T 17

#Chr8 274 275 C A 58

#Chr8 313 314 T C 52

#Chr8 403 404 C A 38

#Chr8 463 464 G A 63

#Chr8 505 506 G A 61

#Chr8 542 543 C T 66

#The output is in a non-standard tab-separated table format, with each row describing a molecule and for each the following columns

#1. MOLECULE\_ID

#2. CHROM

#3. START

#4. END

#5. ALLELE\_STRING{0,<,1,>}

#6. QUAL\_STRING {PHRED+Q33}

#7. #\_OF\_SNPS

#8. #\_OF\_READS

#9. RECOMBINATION\_INTERVALS[HYPHENED INTERVALS,COMMA\_SEPARATED]

#10. POSITION\_OF\_SNPS[COMMA\_SEPARATED]

#11. READS\_IN\_MOLECULE[HYPHENED INTERVALS,COMMA\_SEPARATED]

1. There is an alternative script: bed\_write.pl, which appears to have the same function.
2. ???Why here? It requires vcf!

Phasing HAPCut2, as implemented in pipeline: hapcutVcf.helera1\_demo.sh (example)

#This script is used to create a combined, phased VCF file for a single individual sample

#It does so with the following steps:

#

#1. generate two temporary VCF files - one with only heterozygous sites, and the other one with all sites

#2. HAPCUT2 pipeline, with basically three steps:

#2a. Parse the bam file for reads that correspond to the heterozygous sites for HAPCUT2 using the extractHAIRs utility with the 10X flag on to generate a BX-tagged, unlinked fragment file

#2b. Link the fragments using HAPCUT2's LinkFragments.py utility.

#2c. Run HAPCUT2 proper, with VCF output option

#3. Perform some basic data extraction from the HAPCUT2 output

#4. Merge the full VCF file with the HAPCUT2-derived VCF files and annotating the INFO and FORMAT fields accordingly

#5. Clean-up

From Supplementary: We extracted reads overlapping phase or haplotype-informative positions following the pipeline as outlined by HAPCUT2. These phase-informative molecules were used to determine the number of phase-informative reads per molecule, informative molecule size, haplotype phasing, phase blocks, as well as short and long switch error rates for molecules spanning at least 4 phase-informative SNPs.

1. We then parsed the beadTag output to identify “molecules”. We also followed the definition used by longranger and defined each molecule as a cluster of reads sharing the same beadTag within 50 kbp of each other. We then analyzed the molecules for the SNP alleles and classify them as “concordant” if a given position belongs to the majority allele and otherwise as “discordant” positions. We discarded molecules overlapping 2 or fewer SNPs, and assigned phasing of each molecule to Haplotype 0 or Haplotype 1 if they carry one or no discordant positions. We classified molecules carrying 2 or more discordant positions as “mixed molecules”.
2. Heliconius specific: SNP calling using bcftools call with the multiallelic calling algorithm (-m) 63. The set of BAM files and the raw SNP set were then used as input for the statistical phasing program STITCH 11. Initial parameter tuning were performed to maximize call rate and genotype concordance at focal loci with known genotypes or major colour pattern loci: Chr5:415,36,431 (rs33219710) and 41,536,498 (rs33600994) for the Longshanks mice 13; WntA, optix and Ro for H. erato; and WntA, optix and cortex for H. melpomene (Supplementary Table 8). The colour loci are co-dominant except for Ro in H. erato and cortex in H. melpomene 27.Then these parameters were refined to maximize informativeness (INFO\_SCORE) and SNP diagnostic statistics such as the transition/transversion (TsTv) ratio. Phasing parameters for the different datasets are shown below in their own sections.