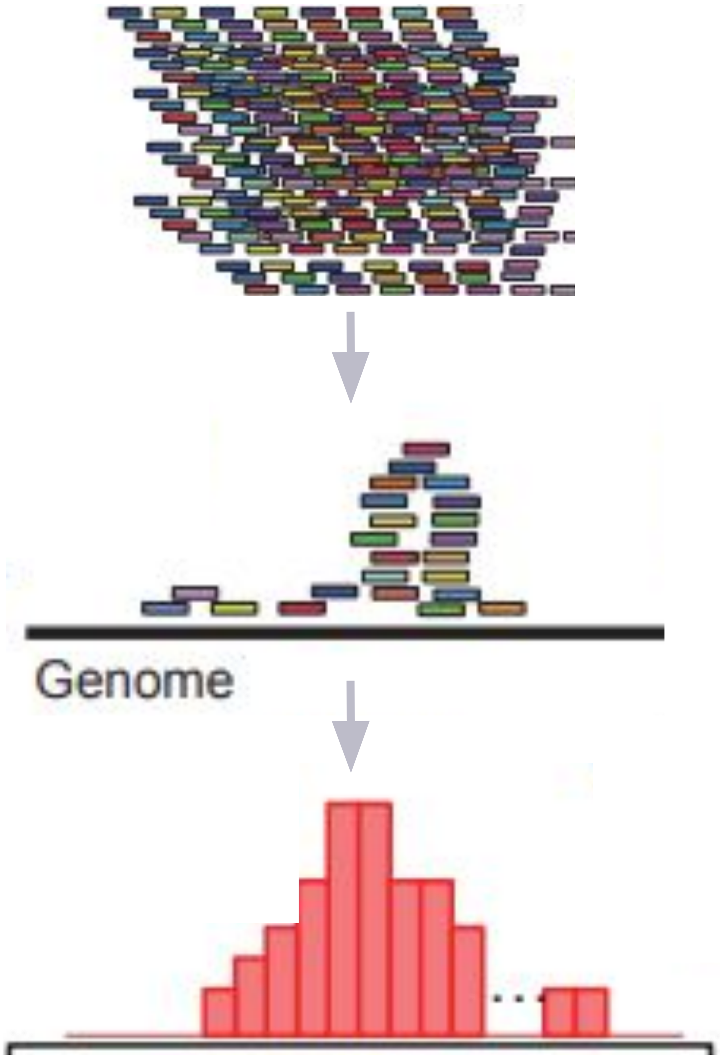
QLS Breakfast Seminar: Sequencing Analysis

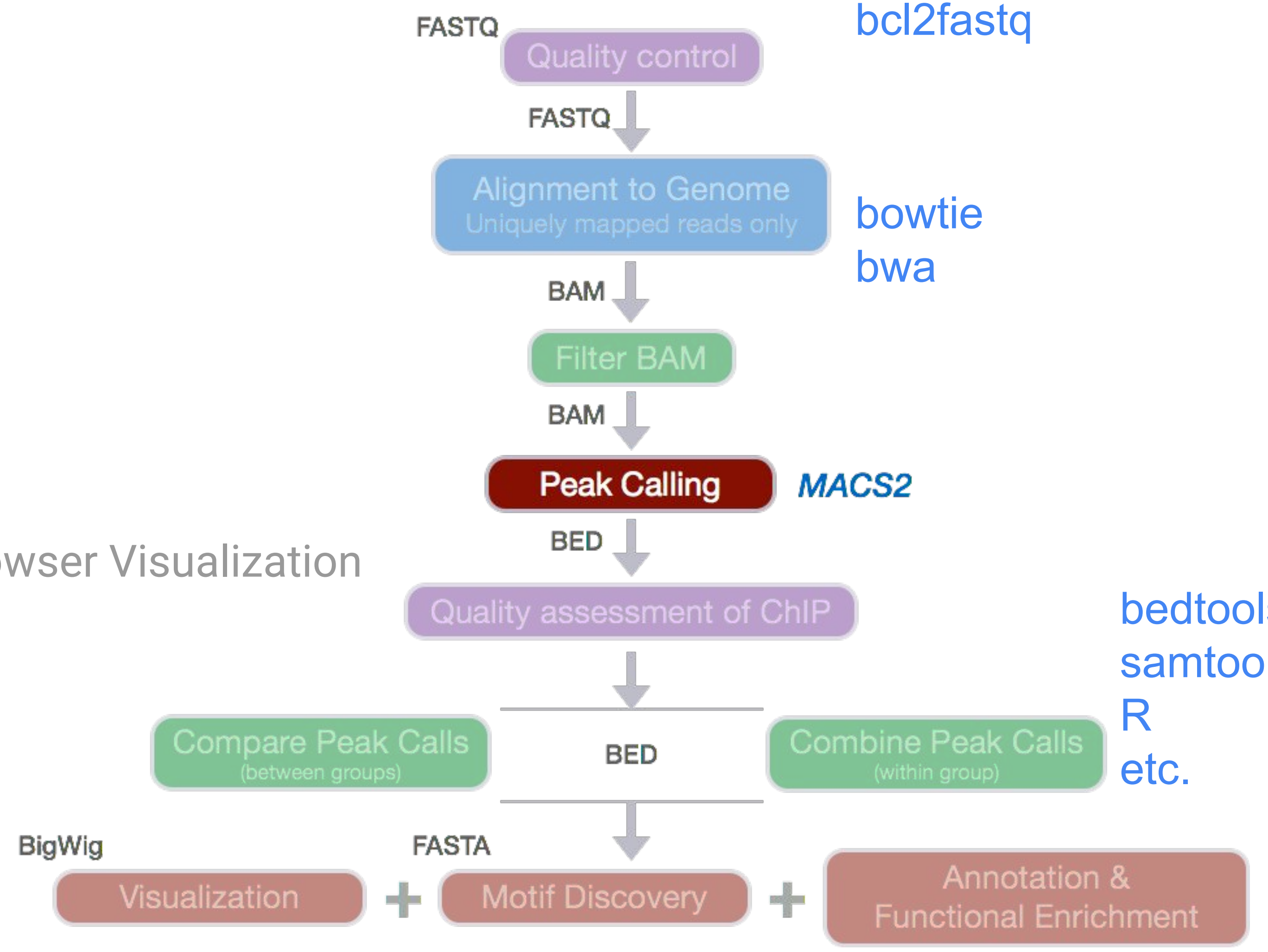
Seth Weaver, Raven Luo-LeBlanc
Craig Lowe’s Vertebrate Genetics Lab

Pipeline


Genome

File Types

Software



bcl2fastq

bowtie
bwa

Tsv
Genome Browser Visualization
etc.

bedtools
samtools
R
etc.

File Types

What does a FASTQ file look like?

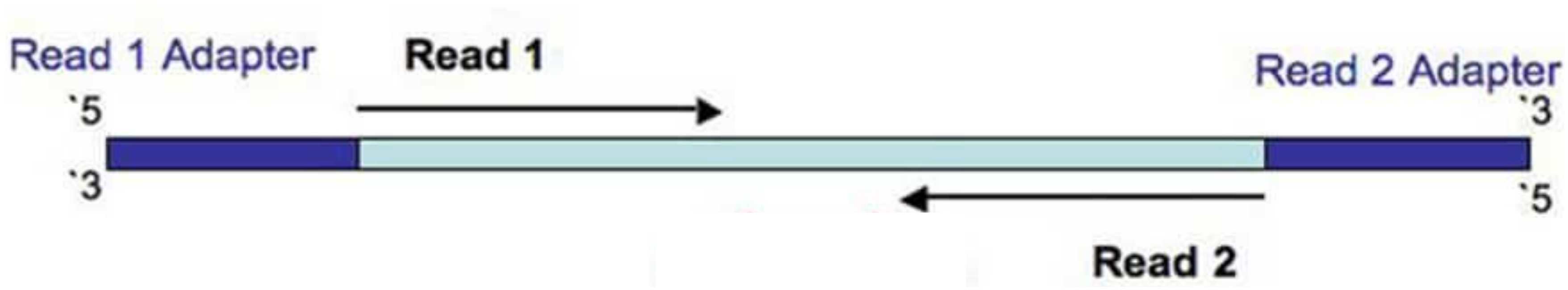
For each cluster that passes filter, a single sequence is written to the corresponding sample's R1 FASTQ file, and, for a paired-end run, a single sequence is also written to the sample's R2 FASTQ file. Each entry in a FASTQ files consists of 4 lines:

1. A sequence identifier with information about the sequencing run and the cluster. The exact contents of this line vary by based on the BCL to FASTQ conversion software used.
2. The sequence (the base calls; A, C, T, G and N).
3. A separator, which is simply a plus (+) sign.
4. The base call quality scores. These are Phred +33 encoded, using ASCII characters to represent the numerical quality scores.

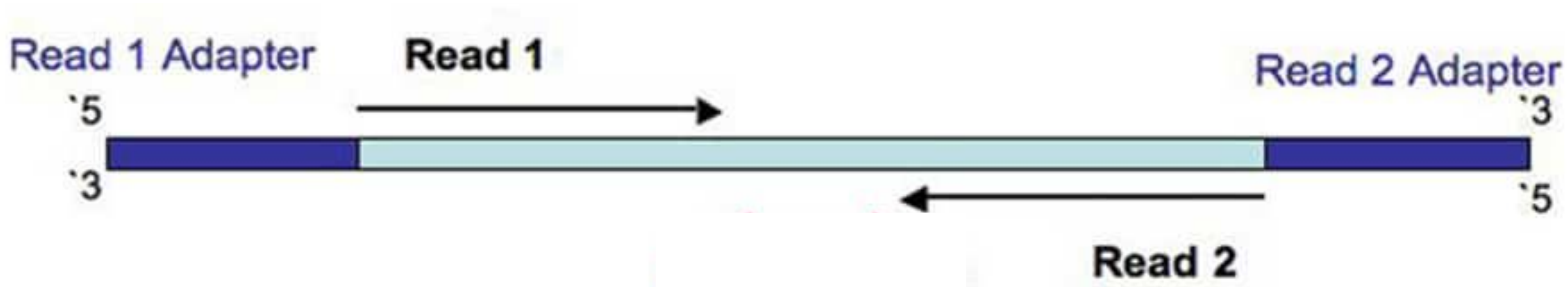
Here is an example of a single entry in a R1 FASTQ file:

```
@ML-P2-14:9:000H003HG:1:11102:17290:1073 1:N:0:TCCTGAGC+GCGATCTA
TTTGGTAACAGCATGAATTATTCTAGCCACTAAAACTCTATGAACATCTTGTGAAGGTTTCAGATAGAGCCTGAAGTACACAGAGAACAATTCTTAAAAAA
+
AAAAAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE<AEEEEEEE
```

Paired end Fastq files



Paired end Fastq files

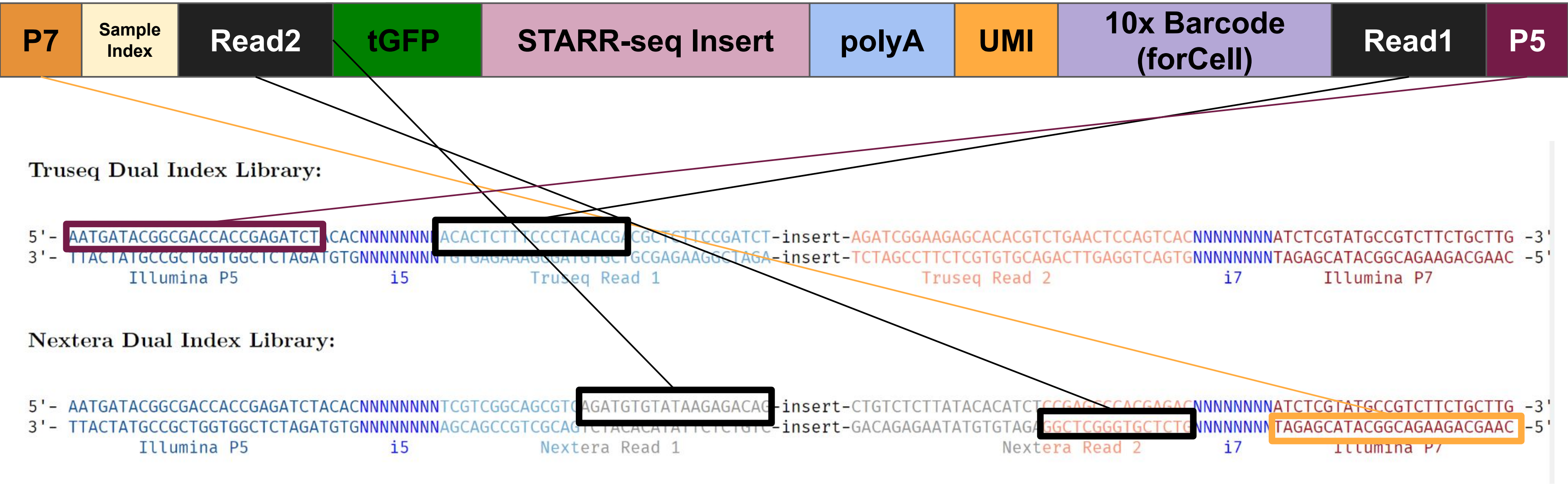


- 2 different fastq files are generated

  __________.R1.fastq   ___________.R2.fastq

- Each pair of fastq reads will share a read name, and be in the same order in their respective fastq files

Final Library Read Structure (Custom, Target Enrichment)



Truseq Dual Index Library:

```
5'- AATGATACGGCGACCACCGAGATCTACACNNNNNNNNACACTCTTTCCCTACACGACGCTCTTCCGATCT-insert-AGATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNNNATCTCGTATGCCGTCTTCTGCTTG -3'
3'- TTACTATGCCGCTGGTGGCTCTAGATGTGNNNNNNNNTGTGAGAAAGGGATGTGCTGCGAGAAGGCTAGA-insert-TCTAGCCTTCTCGTGTGCAGACTTGAGGTCAGTGNNNNNNNNTAGAGCATACGGCAGAAGACGAAC -5'
            Illumina P5               i5          Truseq Read 1                     Truseq Read 2                 i7         Illumina P7
```

Nextera Dual Index Library:

```
5'- AATGATACGGCGACCACCGAGATCTACACNNNNNNNNTCGTCGGCAGCGTCAGATGTGTATAAGAGACAG-insert-CTGTCTCTTATACACATCTCCGAGCCCACGAGACNNNNNNNNATCTCGTATGCCGTCTTCTGCTTG -3'
3'- TTACTATGCCGCTGGTGGCTCTAGATGTGNNNNNNNNAGCAGCCGTCGCAGTCTACACATATTCTCTGTC-insert-GACAGAGAATATGTGTAGAGGCTCGGGTGCTCTGNNNNNNNNTAGAGCATACGGCAGAAGACGAAC -5'
             Illumina P5              i5           Nextera Read 1                 Nextera Read 2              i7          Illumina P7
```

P7 - Nextera sample index i7 - Nextera Read2 Outer - Nextera Read1 Inner - tGFP Inner - STARR-seq Insert - polyA - UMI - 10x barcode - TruSeq Read1 Outer - P5 Outer

Size:
P7 (27bp) + Nextera sample index i7 (8bp) + Nextera Read2 Outer (15bp) + Nextera Read1 Inner (19bp) + "tGFP Inner - STARR-seq Insert - polyA" (37+500+158 bp) (searched tGFP inner primer sequence in SingleCell > Designs > scSS_FoxP2_Vector_Example_Insert.dna, get everything from tGFP inner match to end of SV40 poly(A) signal, but replace Example Insert with 500bp) + UMI (12bp) + 10x barcode (16bp) + TruSeq Read1 Outer (21bp) + P5 Outer (25bp) = 838bp
27 + 8 + 15 + 19 + 37 + 500 + 158 + 12 + 16 + 21 + 25 = 838bp

Example SAM

Example Header Lines

```
@HD     VN:1.0  SO:coordinate
@SQ     SN:1    LN:249250621    AS:NCBI37       UR:file:/data/local/ref/GATK/human_g1k_v37.fasta        M5:1b22b98cdeb4a9304cb5d48026a85128
@SQ     SN:2    LN:243199373    AS:NCBI37       UR:file:/data/local/ref/GATK/human_g1k_v37.fasta        M5:a0d9851da00400dec1098a9255ac712e
@SQ     SN:3    LN:198022430    AS:NCBI37       UR:file:/data/local/ref/GATK/human_g1k_v37.fasta        M5:fdfd811849cc2fadebc929bb925902e5
@RG     ID:UM0098:1     PL:ILLUMINA     PU:HWUSI-EAS1707-615LHAAXX-L001 LB:80   DT:2010-05-05T20:00:00-0400     SM:SD37743      CN:UMCORE
@RG     ID:UM0098:2     PL:ILLUMINA     PU:HWUSI-EAS1707-615LHAAXX-L002 LB:80   DT:2010-05-05T20:00:00-0400     SM:SD37743      CN:UMCORE
@PG     ID:bwa  VN:0.5.4
@PG     ID:GATK TableRecalibration      VN:1.0.3471     CL:Covariates=[ReadGroupCovariate, QualityScoreCovariate, CycleCovariate, DinucCovariate, TileCovariate], default_read_group=null, default_platform=null,
force_read_group=null, force_platform=null, solid_recal_mode=SET_Q_ZERO, window_size_nqs=5, homopolymer_nback=7, exception_if_no_tile=false, ignore_nocall_colorspace=false, pQ=5, maxQ=40, smoothing=1
```

In the alignment examples below, you will see that the 2nd alignment maps back to the RG line with ID UM0098.1, and all of the alignments point back to the SQ line with SN:1 because their RNAME is 1.

Example Alignments

This is what the alignment section of a SAM file looks like:

```
1:497:R:-272+13M17D24M  113     1       497     37      37M     15      100338662       0       CGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAG   0;==-==9;>>>>>=>>>>>>>>>>>>=>>>>>>>>>   XT:A:U  NM:i:0  SM:i:37 AM:i:0  X0:i:1  X1:i:0
XM:i:0  XO:i:0  XG:i:0  MD:Z:37
19:20389:F:275+18M2D19M 99      1       17644   0       37M     =       17919   314     TATGACTGCTAATAATACCTACACATGTTAGAACCAT   >>>>>>>>>>>>>>>>>>>><<>>><<>>4::>>:<9   RG:Z:UM0098:1   XT:A:R  NM:i:0  SM:i:0  AM:i:0  X0:i:4
X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:37
19:20389:F:275+18M2D19M 147     1       17919   0       18M2D19M        =       17644   -314    GTAGTACCAACTGTAAGTCCTTATCTTCATACTTTGT   ;44999;499<8<8<<<8<<><<<<><7<;<<<>><<   XT:A:R  NM:i:2  SM:i:0  AM:i:0  X0:i:4  X1:i:0
XM:i:0  XO:i:1  XG:i:2  MD:Z:18^CA19
9:21597+10M2I25M:R:-209 83      1       21678   0       8M2I27M =       21469   -244    CACCACATCACATATACCAAGCCTGGCTGTGTCTTCT   <;9<<5><<<<><<<>><<><>><9>><>>>9>>><>   XT:A:R  NM:i:2  SM:i:0  AM:i:0  X0:i:5  X1:i:0  XM:i:0
XO:i:1  XG:i:2  MD:Z:35
```

In this example, the fields are:

Field	Alignment 1	Alignment 2	Alignment 3	Alignment 4
QNAME	1:497:R:-272+13M17D24M	19:20389:F:275+18M2D19M	19:20389:F:275+18M2D19M	9:21597+10M2I25M:R:-209
FLAG	113	99	147	83
RNAME	1	1	1	1
POS	497	17644	17919	21678
MAPQ	37	0	0	0
CIGAR	37M	37M	18M2D19M	8M2I27M
MRNM/RNEXT	15	=	=	=
MPOS/PNEXT	100338662	17919	17644	21469
ISIZE/TLEN	0	314		
SEQ	CGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAG	TATGACTGCTAATAATACCTACACATGTTAGAACCAT	GTAGTACCAACTGTAAGTCCTTATCTTCATACTTTGT	CACCACATCACATATACCAAGCCTGGCTGTGTCTTCT
QUAL	0;==-==9;>>>>>=>>>>>>>>>>>>=>>>>>>>>>	>>>>>>>>>>>>>>>>>>>><<>>><<>>4::>>:<9	;44999;499<8<8<<<8<<><<<<><7<;<<<>><<	<;9<<5><<<<><<<>><<><>><9>><>>>9>>><>
TAGs	XT:A:U NM:i:0 SM:i:37 AM:i:0 X0:i:1 X1:i:0 XM:i:0 XO:i:0 XG:i:0 MD:Z:37	RG:Z:UM0098:1 XT:A:R NM:i:0 SM:i:0 AM:i:0 X0:i:4 X1:i:0 XM:i:0 XO:i:0 XG:i:0 MD:Z:37	XT:A:R NM:i:2 SM:i:0 AM:i:0 X0:i:4 X1:i:0 XM:i:0 XO:i:1 XG:i:2 MD:Z:18^CA19	XT:A:R NM:i:2 SM:i:0 AM:i:0 X0:i:5 X1:i:0 XM:i:0 XO:i:1 XG:i:2 MD:Z:35

What is a CIGAR?

You may have heard the term CIGAR, but wondered what it means. Hopefully this section will help clarify it.

The sequence being aligned to a reference may have additional bases that are not in the reference or may be missing bases that are in the reference. The CIGAR string is a sequence of of base lengths and the associated operation. They are used to indicate things like which bases align (either a match/mismatch) with the reference, are deleted from the reference, and are insertions that are not in the reference.

For example:

```
RefPos:     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
Reference:  C  C  A  T  A  C  T  G  A  A  C  T  G  A  C  T  A  A  C
Read: ACTAGAATGGCT
```

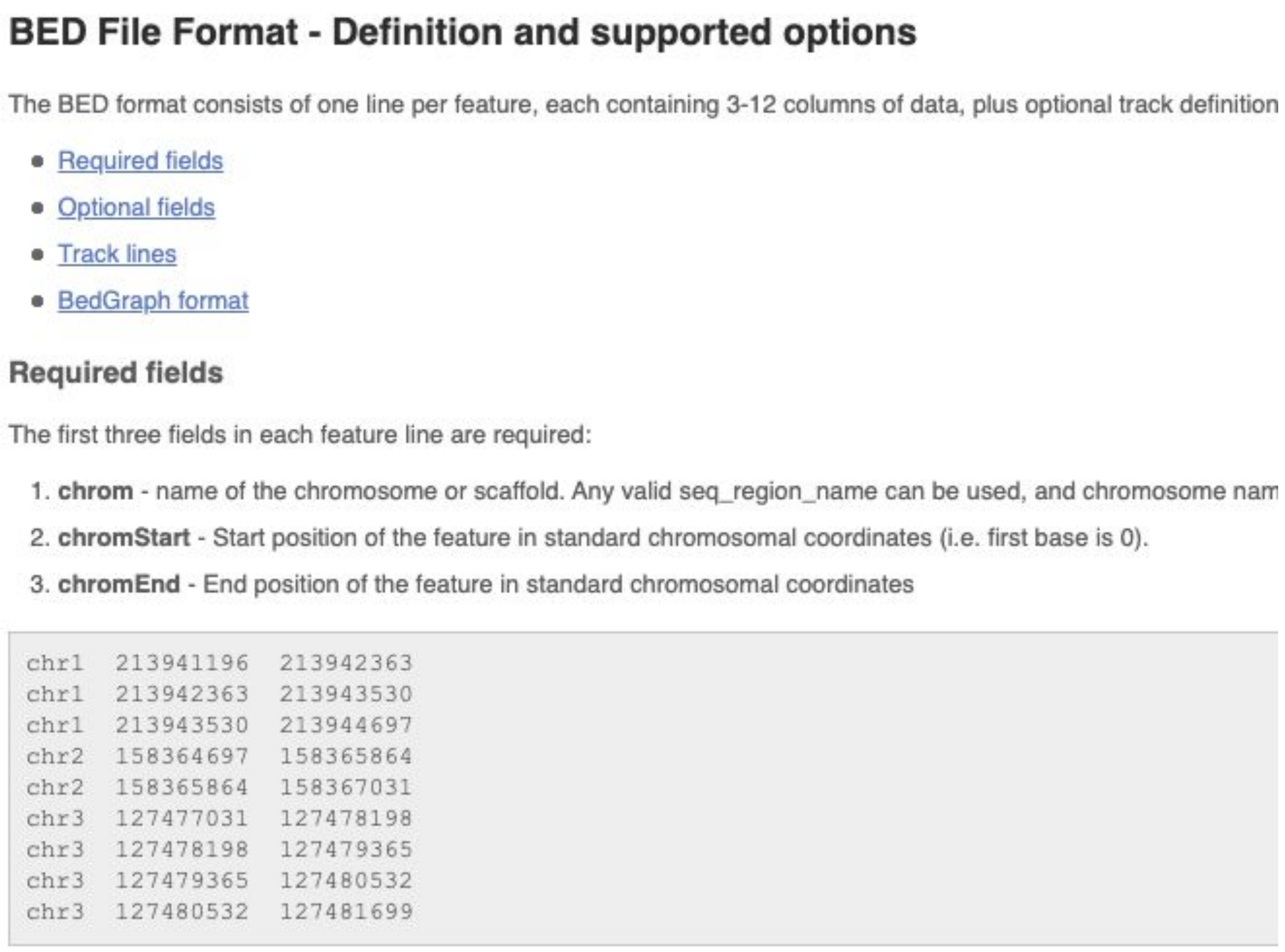
Aligning these two:

```
RefPos:     1  2  3  4  5  6  7     8  9 10 11 12 13 14 15 16 17 18 19
Reference:  C  C  A  T  A  C  T     G  A  A  C  T  G  A  C  T  A  A  C
Read:                   A  C  T  A  G  A  A     T  G  G  C  T
```

With the alignment above, you get:

```
POS: 5
CIGAR: 3M1I3M1D5M
```

The POS indicates that the read aligns starting at position 5 on the reference. The CIGAR says that the first 3 bases in the read sequence align with the reference. The next base in the read does not exist in the reference. Then 3 bases align with the reference. The next reference base does not exist in the read sequence, then 5 more bases align with the reference. Note that at position 14, the base in the read is different than the reference, but it still counts as an M since it aligns to that position.



BED File Format - Definition and supported options

The BED format consists of one line per feature, each containing 3-12 columns of data, plus optional track definition

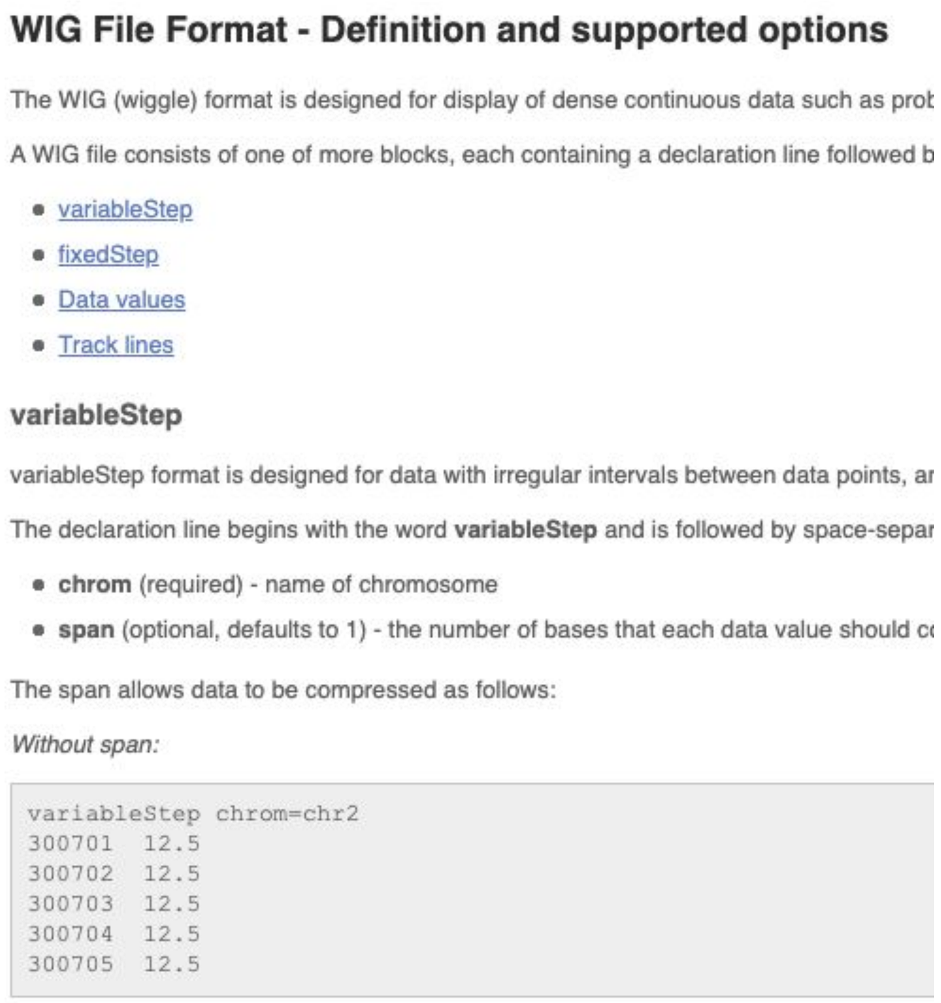
- Required fields
- Optional fields
- Track lines
- BedGraph format

Required fields

The first three fields in each feature line are required:

1. **chrom** - name of the chromosome or scaffold. Any valid seq_region_name can be used, and chromosome nam
2. **chromStart** - Start position of the feature in standard chromosomal coordinates (i.e. first base is 0).
3. **chromEnd** - End position of the feature in standard chromosomal coordinates

```
chr1  213941196  213942363
chr1  213942363  213943530
chr1  213943530  213944697
chr2  158364697  158365864
chr2  158365864  158367031
chr3  127477031  127478198
chr3  127478198  127479365
chr3  127479365  127480532
chr3  127480532  127481699
```



WIG File Format - Definition and supported options

The WIG (wiggle) format is designed for display of dense continuous data such as prob

A WIG file consists of one of more blocks, each containing a declaration line followed b

- variableStep
- fixedStep
- Data values
- Track lines

variableStep

variableStep format is designed for data with irregular intervals between data points, ar

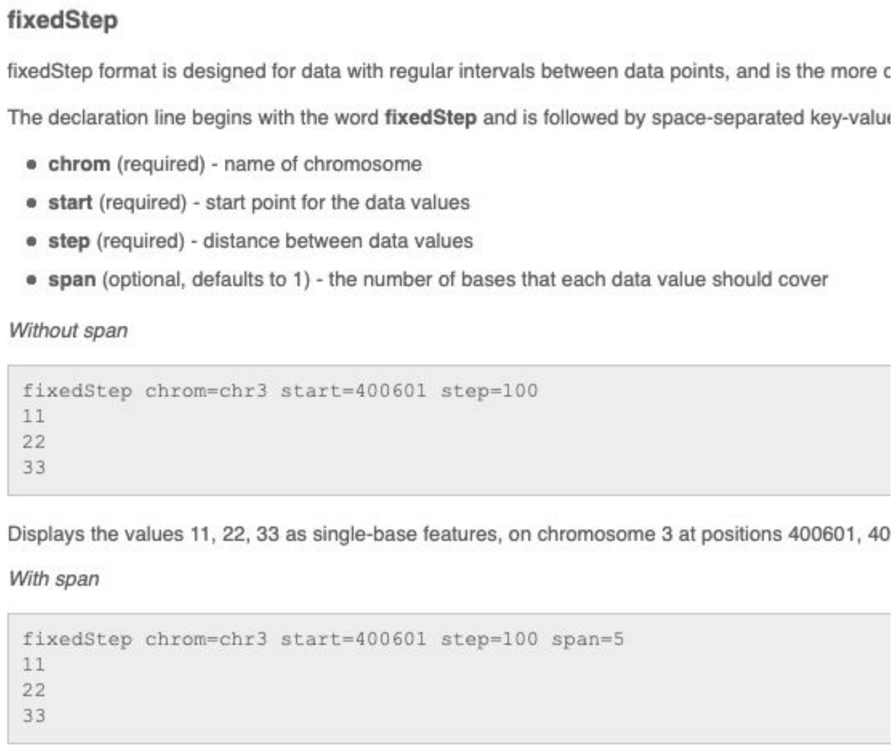
The declaration line begins with the word **variableStep** and is followed by space-separ

- **chrom** (required) - name of chromosome
- **span** (optional, defaults to 1) - the number of bases that each data value should co

The span allows data to be compressed as follows:

Without span:

```
variableStep chrom=chr2
300701  12.5
300702  12.5
300703  12.5
300704  12.5
300705  12.5
```



fixedStep

fixedStep format is designed for data with regular intervals between data points, and is the more c

The declaration line begins with the word **fixedStep** and is followed by space-separated key-valu

- **chrom** (required) - name of chromosome
- **start** (required) - start point for the data values
- **step** (required) - distance between data values
- **span** (optional, defaults to 1) - the number of bases that each data value should cover

Without span

```
fixedStep chrom=chr3 start=400601 step=100
11
22
33
```

Displays the values 11, 22, 33 as single-base features, on chromosome 3 at positions 400601, 40

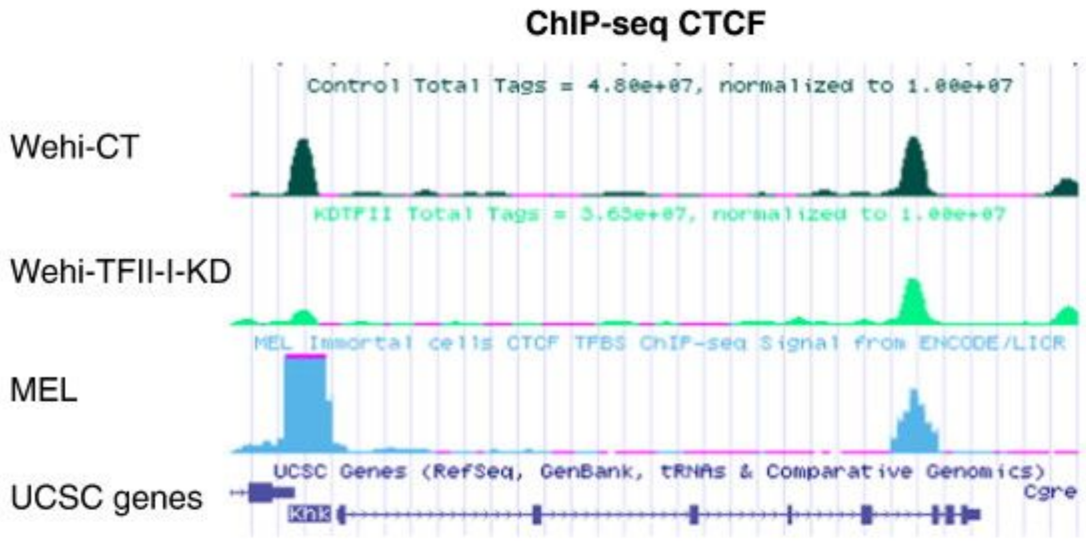
With span

```
fixedStep chrom=chr3 start=400601 step=100 span=5
11
22
33
```

FASTA format example

```
>MyGene
CCTCTCGGAGCTGGAAATGCAGCTATTGAGATCTTCGAATGCTGCGGAGCTGGAGGCGGA
GGCAGCTGGGGAGGTCCGAGCGATGTGACCAGGCCGCCATCGCTCGTCTCTTCCTCTCTC
CTGCCGCCTCCTGTGTCGAAAATAACTTTTTTAGTCTAAAGAAAGAAAGACAAAAGTAGT
CGTCCGCCCCTCACGCCCTCTCTTCCTCTCAGCCTTCCGCCCGGTGAGGAAGCCCGGGGT
GGCTGCTCCGCCGTCGGGGCCGCGCCGCCGAGCCCCAGCGCCCCGGGCCGCCCCCGCACG
CCGCCCCCATGCATCCCTTCTACACCCGGGCCGCCACCATGATAGGCGAGATCGCCGCCG
CCGTGTCCTTCATCTCCAAGTTTCTCCGCACCAAGGGGCTGACGAGCGAGCGACAGCTGC
AGACCTTCAGCCAGAGCCTGCAGGAGCTGCTGGCAGAACATTATAAACATCACTGGTTCC
CAGAAAAGCCATGCAAGGGATCGGGTTACCGTTGTATTCGCATCAACCATAAAATGGATC
```

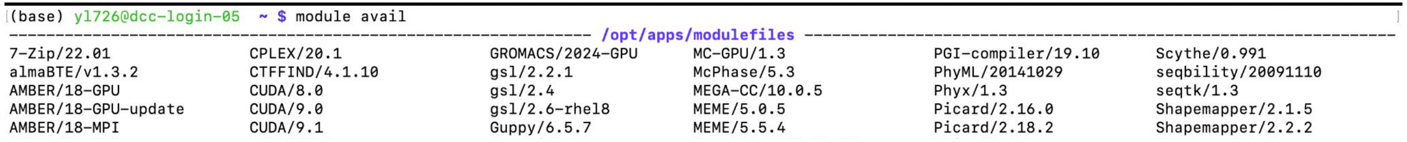
Genome Browser Visualization



Software

How to use software

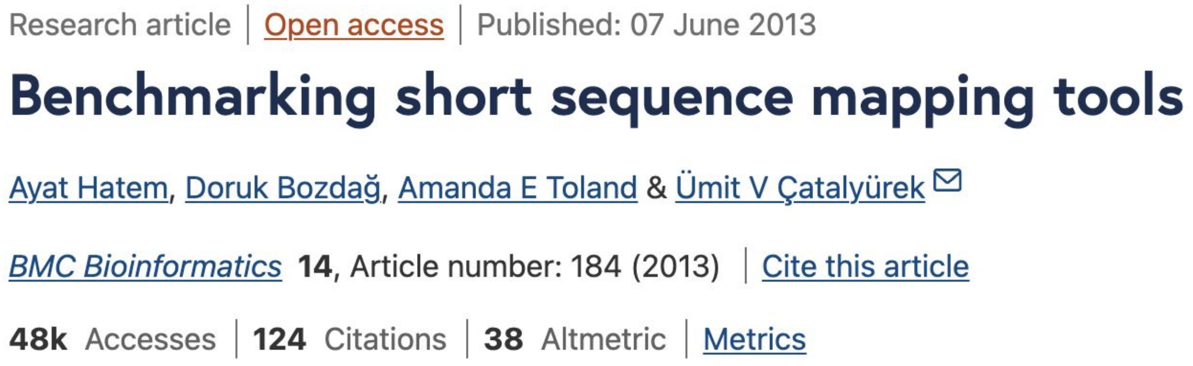
- Bcl2fastq: DCC module or self install, command line
- Bowtie: DCC module, command line
- BWA: DCC module, command line
- MACS: DCC install miniconda, make miniconda virtual Python env, activate env, install macs3, command line
- Samtools: DCC module, command line
- Bedtools: DCC module, command line
- R: DCC module or self install with RStudio, command line or GUI with RStudio
- Genome Browser: website, GUI



```
(base) yl726@dcc-login-05  ~ $ module avail
------------------------------------------------------------- /opt/apps/modulefiles ----------------------------------------------------------------
7-Zip/22.01            CPLEX/20.1             GROMACS/2024-GPU    MC-GPU/1.3             PGI-compiler/19.10    Scythe/0.991
almaBTE/v1.3.2         CTFFIND/4.1.10         gsl/2.2.1           McPhase/5.3            PhyML/20141029        seqbility/20091110
AMBER/18-GPU           CUDA/8.0               gsl/2.4             MEGA-CC/10.0.5         Phyx/1.3              seqtk/1.3
AMBER/18-GPU-update    CUDA/9.0               gsl/2.6-rhel8       MEME/5.0.5             Picard/2.16.0         Shapemapper/2.1.5
AMBER/18-MPI           CUDA/9.1               Guppy/6.5.7         MEME/5.5.4             Picard/2.18.2         Shapemapper/2.2.2
```

Different software for the same step, e.g. for alignment, bowtie vs bwa

- "However, Bowtie maintained the best throughput for most of the tests while BWA performed better for longer read lengths."
- Bowtie: ChIP-seq, ATAC-seq
- Bwa: RNA-seq (STAR may be better due to splice awareness)



Research article | Open access | Published: 07 June 2013

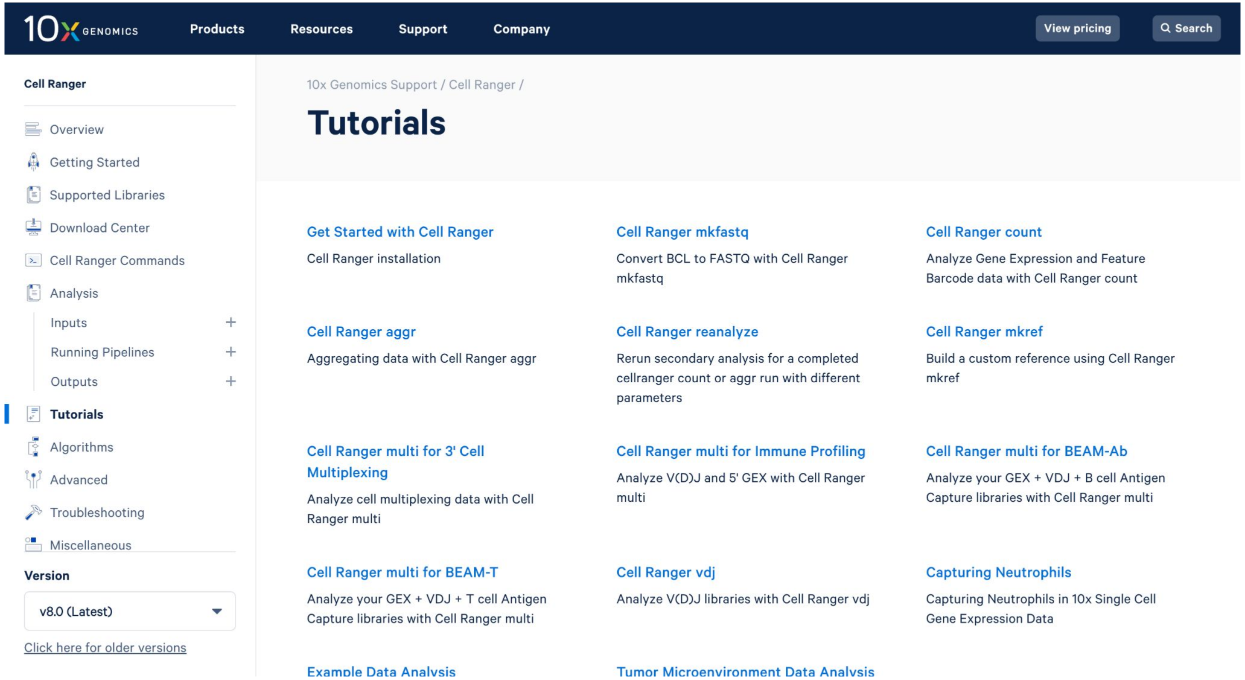
Benchmarking short sequence mapping tools

Ayat Hatem, Doruk Bozdağ, Amanda E Toland & Ümit V Çatalyürek

BMC Bioinformatics **14**, Article number: 184 (2013) | Cite this article

48k Accesses | **124** Citations | **38** Altmetric | Metrics

Single-cell, e.g. cellranger



Cell Ranger

- Overview
- Getting Started
- Supported Libraries
- Download Center
- Cell Ranger Commands
- Analysis
  - Inputs +
  - Running Pipelines +
  - Outputs +
- **Tutorials**
- Algorithms
- Advanced
- Troubleshooting
- Miscellaneous

Version

v8.0 (Latest)

Click here for older versions

10x Genomics Support / Cell Ranger /

Tutorials

Get Started with Cell Ranger
Cell Ranger installation

Cell Ranger mkfastq
Convert BCL to FASTQ with Cell Ranger mkfastq

Cell Ranger count
Analyze Gene Expression and Feature Barcode data with Cell Ranger count

Cell Ranger aggr
Aggregating data with Cell Ranger aggr

Cell Ranger reanalyze
Rerun secondary analysis for a completed cellranger count or aggr run with different parameters

Cell Ranger mkref
Build a custom reference using Cell Ranger mkref

Cell Ranger multi for 3' Cell Multiplexing
Analyze cell multiplexing data with Cell Ranger multi

Cell Ranger multi for Immune Profiling
Analyze V(D)J and 5' GEX with Cell Ranger multi

Cell Ranger multi for BEAM-Ab
Analyze your GEX + VDJ + B cell Antigen Capture libraries with Cell Ranger multi

Cell Ranger multi for BEAM-T
Analyze your GEX + VDJ + T cell Antigen Capture libraries with Cell Ranger multi

Cell Ranger vdj
Analyze V(D)J libraries with Cell Ranger vdj

Capturing Neutrophils
Capturing Neutrophils in 10x Single Cell Gene Expression Data

Example Data Analysis

Tumor Microenvironment Data Analysis

Applications of sequencing analysis

Poll: what sequencing data are you hoping to analyse?

- Sequencing data that you generated
- Sequencing data that a labmate or collaborator generated
- Publically available sequencing data from a consortium or publication?
- None right now, just hoping to learn more

What can you do with NGS sequencing data?

1. Look for enrichments/pile up of reads
    a. ATAC-seq, ChIP-seq, DNase-seq, etc… (find regulatory elements)
2. Find mutations in samples compared to a reference
    a. Whole genome/Exome sequencing
3. Look at the transcriptome, find differentially expressed genes between conditions
    a. RNA-seq
4. Visualize on genome browser

Many, many more….

What can you do with NGS sequencing data?

- NGS creates a lot of data, more that is often talked about in a publication. If you see a publications that has done an interesting sequencing experiment in a relevant cell type/tissue to you, download and reanalyze for things you are interested in!

Fastq alignment with BWA. Tutorial on DCC

You can copy the whole tutorial over to your directory in DCC with:

```
cp -r /hpc/group/vertgenlab/seth/qlsBreakfast/ /path/to/your/dir/
```

Feel free to reach out to me for any sequencing questions:

seth.weaver@duke.edu
yanting.luo@duke.edu