Population genomics with sequenced pools (Pool-Seq) - 1

Dr. Robert Kofler

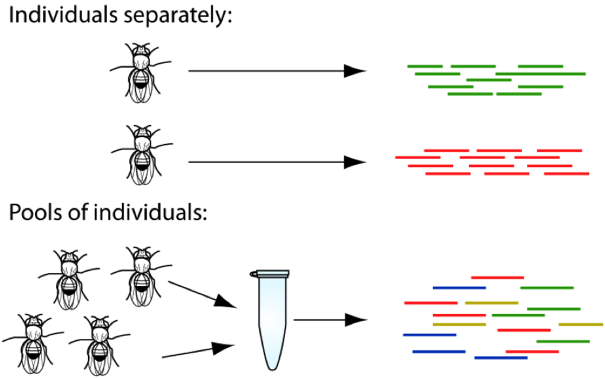
February 14, 2014

Availability of slides

http://drrobertkofler.wikispaces.com/PoPoolationGenomics

TWO SEQUENCING STRATEGIES?

Population genomics requires sequencing of genomic data:



Individuals separately:

Pools of individuals:

Pooling: pros and cons

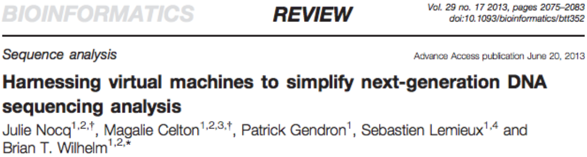
- +++ cost effective; a single Illumina lane may be sufficient to estimate allele frequencies for one population
- + bioinformatics analysis is, in my opinion, simpler as one population is represented by one sample. You do not have to deal with many individual sequences per population.
- - haplotype information is lost
- - distinguishing between minor alleles and sequencing errors may be very difficult
- sometimes it may simple not be possible to sequence individuals, e.g.: cancer samples are mostly sequenced as pools

What could be done with pools?

- Estimate polymorphism in natural populations (PoPoolation1)
- Identify signatures of positive selection (PoPoolation1)
- Estimate differentiation between natural populations (PoPoolation2)
- Pool-GWAS: phenotypically extreme individuals (e.g.: white and black abdomen) are separated into distinct groups and these groups are sequenced as pools. Causative variants (e.g.: pigmentation genes) are identified by contrasting the allele frequencies between the groups (PoPoolation2)
- Experimental evolution: trace the allele frequency of beneficial alleles during adaptation (PoPoolation2)
- Identify the insertion sites and the population frequency of transposable elements (PoPoolation TE)

Virtual Machine

The course will be held on a Virtual Machine. Advantages?



BIOINFORMATICS **REVIEW** *Vol. 29 no. 17 2013, pages 2075–2083*
doi:10.1093/bioinformatics/btt352

Sequence analysis Advance Access publication June 20, 2013

Harnessing virtual machines to simplify next-generation DNA sequencing analysis

Julie Nocq[1,2,†], Magalie Celton[1,2,3,†], Patrick Gendron[1], Sebastien Lemieux[1,4] and Brian T. Wilhelm[1,2,*]

- correct versions of software and all required libraries are preinstalled so you may actually be able to repeat the demonstrated analysis
- reproducibility of analysis is increased; data may be shared together with the software necessary for analysing them
- more stable pipelines; entire pipelines may be shared
- this comes at a cost: performance loss of about 25%

A MORE PERSONAL ACCOUNT

I already taught several courses, and never, not even in our own Institute did we manage to teach such a course without major software problems. Either were some tools missing or the wrong versions where installed. Previously I was however only teaching PoPoolation1, this time I'm teaching PoPoolation1, PoPoolation2 and Gowinda. So I had this vision in my minding, me teaching this course....and after about 4 hours.... no one was able to just map the first file. During one of such visions, utterly drenched in sweat, I had the idea that there may be one solution, so that you may actually be able to repeat the analysis: Virtual Machines

PoPoolation 1

OPEN ACCESS Freely available online

PLoS one

PoPoolation: A Toolbox for Population Genetic Analysis of Next Generation Sequencing Data from Pooled Individuals

Robert Kofler[1¶], Pablo Orozco-terWengel[1¶], Nicola De Maio[1], Ram Vinay Pandey[1], Viola Nolte[1], Andreas Futschik[2], Carolin Kosiol[1], Christian Schlötterer[1*]

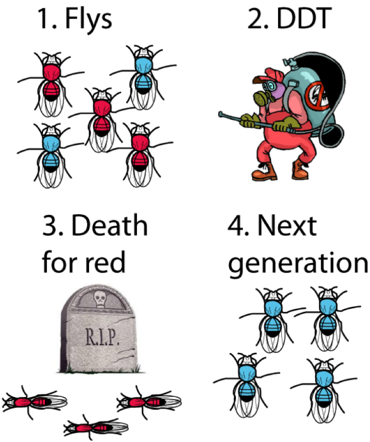
[1] Institute of Population Genetics, Vetmeduni Vienna, Vienna, Austria, [2] Department of Statistics, University of Vienna, Vienna, Austria

WHAT CAN YOU DO WITH POPOOLATION?

- Perform genome-wide scans for positively selected regions in populations sequenced as pools
- Obtain genome-wide estimates of natural variation
- You may just estimate natural variation at synonymous or non-synonymous sites
- trim fastq-reads

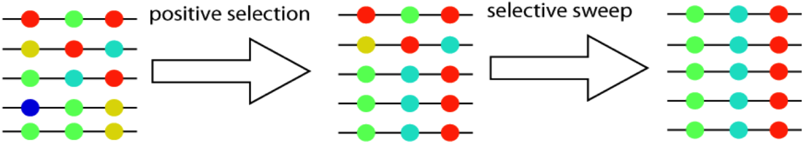
POSITIVE SELECTION?

Individuals with beneficial mutations (e.g.: resistance to DDT) will produce more progeny, and thus the beneficial allele rises in frequency.



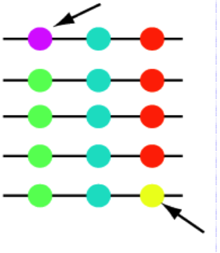
GENOMIC SIGNATURE OF POSITIVE SELECTION #1

When an allele increases in its population frequency, nearby variants also increase in its frequency ⇒ Hitchhiking. This leads to a selective sweep which erases variation around a positively selected allele.



GENOMIC SIGNATURE OF POSITIVE SELECTION #2

After the sweep new mutations appear and restore diversity, but they appear very slowly (mutations are rare) and they are initially of low frequency



Selective sweeps thus lead to regions with reduced variability in the neighborhood of the selected sites, i.e.: few SNPs having low population frequencies.

EXAMPLE: KEL-LOCUS IN HUMANS

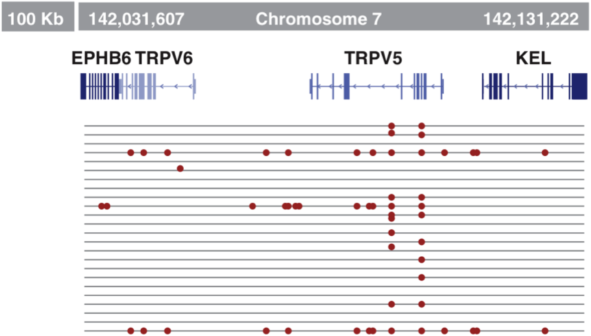


Fig. 3. Low diversity and many rare alleles at the Kell blood antigen cluster. On the basis of three different statistical tests, the 115-kb region (containing four genes) shows evidence of a selective sweep in Europeans (*28*).

KEL extends for 115kb and is the largest locus of positive selection described in humans. This genes are important determinants of blood type.

TAJIMAS π

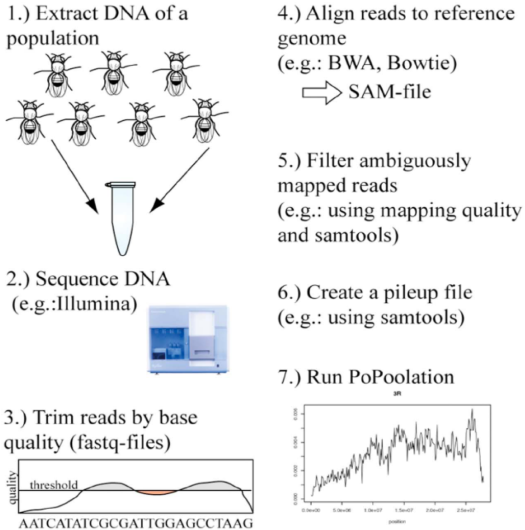
One measure of variability within populations is Tajima's π, which is defined as the average pairwise difference between randomly chosen individuals. Can be calculated for SNPs, windows, genes, etc...

$$\pi = \frac{\sum\limits_{SNPs}(1 - \sum\limits_{alleles} f_a^2)}{L}$$

- f_a frequency of the given allele
- L length of the investigated window

Value of Tajima's π varies between $0 - 1$, where a low π indicates no natural variation (few SNPs with low population frequencies) and a high π a large amount of natural variation (many SNPs with very balanced allele frequencies). In this Walkthrough I will show how to obtain genome-wide estimates of Tajima's π with PoPoolation.

WORKFLOW WITH POPOOLATION



1.) Extract DNA of a population

2.) Sequence DNA (e.g.:Illumina)

3.) Trim reads by base quality (fastq-files)

4.) Align reads to reference genome (e.g.: BWA, Bowtie) ⇨ SAM-file

5.) Filter ambiguously mapped reads (e.g.: using mapping quality and samtools)

6.) Create a pileup file (e.g.: using samtools)

7.) Run PoPoolation

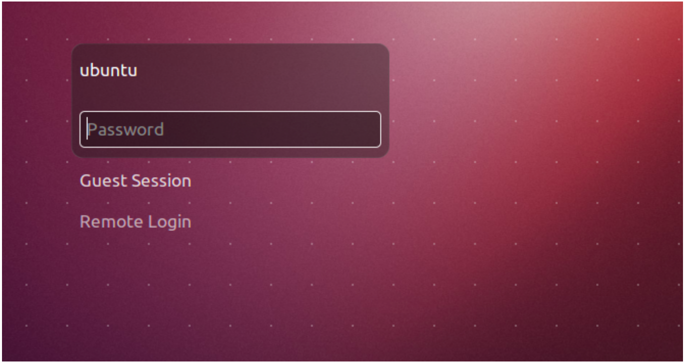
Bioinformatics workflow

As I have here sufficient time (1 day), I will demonstrate the fully optimized pipeline that we use in the Institute of Population Genetics in Vienna. We use this pipeline for our most recent publications. This pipeline includes the following steps:

- trimming of reads
- mapping of reads
- removing duplicates
- removing low quality sequences (ambiguous mapping)
- converting to mpileup
- subsampling to uniform coverage
- remove regions neighboring to indels
- PoPoolation

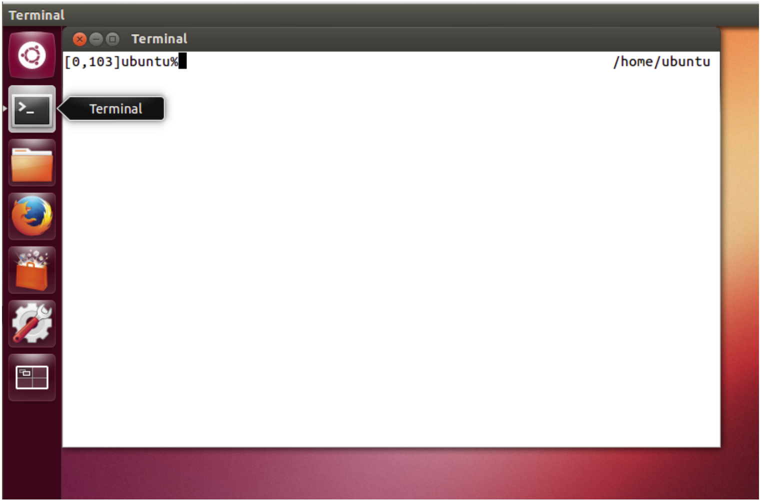
PoPoolation VS

PoPoolation and dependencies installed on a Virtual server; Start the virtual machine



User ID: ubuntu - PWD: reverse

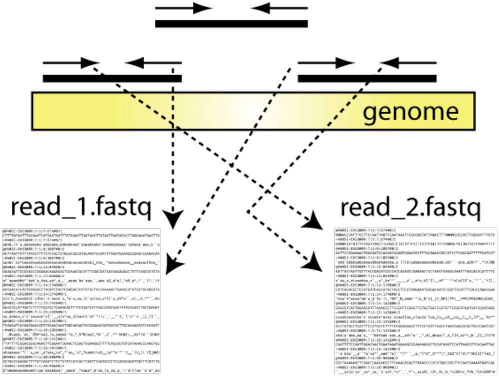
OPEN TERMINAL



```
[0,103]ubuntu%                                                /home/ubuntu
```

Paired end reads

We start with paired end reads. With the Illumina technology you will get two fastq-files for paired end reads. Reads are always provided in the same order in both fastq-files. The two reads of one pair can therefore be recognized by having the same index in the both fastq-files.



Note: the first read (read_1.fastq) is not necessarily the 5′ read. Assignment as the first read is a stochastic process (therefore usually 50% 5′ reads and 50% 3′ reads).

FIRST STEP

ALWAYS make sure your data are complete. Quick and dirty

```
cd Desktop/popoolation1
wc read_*
>  209288  209288 11279782 read_1.fastq
>  209288  209288 11279782 read_2.fastq
>  418576  418576 22559564 total
```

more professional

```
md5sum read_*
>MD5 (read_1.fastq) = fd8fdfce336391e106fdc84ee60dd622
>MD5 (read_2.fastq) = 18d01db158ea29334d90b68880d9f6bb
```

The wordcount or the md5 sum should be compared to the values provided by the sequencing facility (e.g.: BGI).

INSPECTING THE FASTQ FILES

```
less read_1.fastq
```

```
@HWUSI-EAS300R:7:1:7:674#0/1
CTTTTGTAGTTTACAAATCATGAATAATTTATAGAGTTAGTAACTTATAATTAATATACCTAGGAAATAAGTTA
+HWUSI-EAS300R:7:1:7:674#0/1
abbb_[V`a_abaababa`abbaaba_bbbabbaab`aaaabaaba`baabbbabaaa`\bbaaa`aaa_b``a
@HWUSI-EAS300R:7:1:9:1897#0/1
ACTTAATTATTTATGCTTTTCTCTACTCTGCACGGCATGCAAATGCAATATAGATGCAAGGCGAGCCGAAACAA
+HWUSI-EAS300R:7:1:9:1897#0/1
aaab\`bb^bababbaababbbbbababbbbabaaaaabbbb[_bba_`^aabaabaaaa__aaaaaa[Saa_`
@HWUSI-EAS300R:7:1:13:849#0/1
GAAATATTGCGTAGCCGGAAACAAAAGAGTGCAAATACATTTCGACGATGATGAGAGAGCTATTCAGGGCTGTA
+HWUSI-EAS300R:7:1:13:849#0/1
a^`aaaabbb^`bab`a_Xaa_aa^_a__`_aaaa`ba`aaa_`_aaa`aZ_a^a]_^aR_a^_^`_`]^_`\Y
......
```

STRUCTURE OF A FASTQ-FILE

```
@HWUSI-EAS300R:7:1:7:674#0/1
CTTTTGTAGTTTACAAATCATGAATAATTTATAGAGTTAGTAACTTATAATTAATATACCTAGGAAATAAGTTA
+HWUSI-EAS300R:7:1:7:674#0/1
abbb_[V`a_abaababa`abbaaba_bbbabbaab`aaaabaaba`baabbbabaaa`\bbaaa`aaa_b``a
```

Every read occupies four lines in a fastq file

- @: name of the read (always starts with '@')
- CTT..: the sequence of the read
- +: the name again (only useful with multiline fastq entries)
- the base quality of the DNA sequence; one character for every nucleotide

NAVIGATING A FASTQ-FILE

- ↑ ↓ scroll through the file
- ␣ next page
- 'b' previous page
- 'g' move to beginning of file
- 'G' move to end of file
- /blabla search for blabla
- 'n' move to the next search result
- 'N' move to the previous search result

FASTQ QUALITY ENCODING

```
@HWUSI-EAS300R:7:1:7:674#0/1
CTTTTGTAGTTTACAAATCATGAATAATTTATAGAGTTAGTAACTTATAATTAATATACCTAGGAAATAAGTTA
+HWUSI-EAS300R:7:1:7:674#0/1
abbb_[V`a_abaababa`abbaaba_bbbabbaab`aaaabaaba`baabbbabaaa`\bbaaa`aaa_b``a
```

In a fastq file the first quality character refers to the first DNA character and so on..

```
CTTTTGTAGTTTACAAATCATGAATAATTTATAGAGTTAGTAACTTATAATTAATATACCTAGGAAATAAGTTA
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
abbb_[V`a_abaababa`abbaaba_bbbabbaab`aaaabaaba`baabbbabaaa`\bbaaa`aaa_b``a
```

So what does a quality of 'a' or 'b' mean? Usually there would be a simple answer but thanks to Illumina the answer is: "it depends on the quality encoding". Accordingly we now have different types of fastq files.

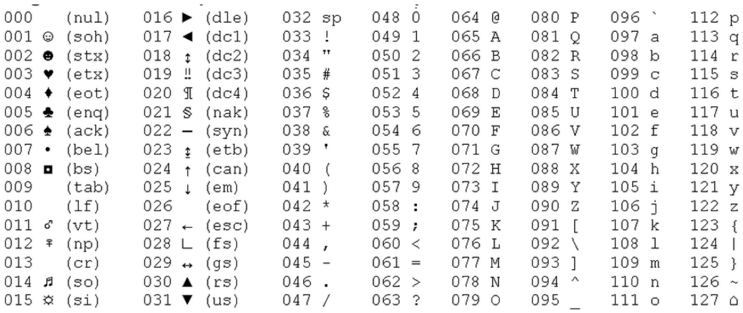
QUALITY DEMYSTIFIED

Basically every base quality entry gives you the probability that the corresponding DNA nucleotide is wrong (thus a sequencing error). So a quality of 0.001 means that on average 1 in 1000 bases having this quality is wrong.
The encoding of the quality involves a few steps

- get the decimal value of the quality character from an ASCII table (see next page)
- subtract the offset (sanger=33 illumina=64)
- voila: the quality of the base; usually a number between 0-40 [=dec(ascii)]
- if you want, you can go on and calculate the probability of being a sequencing error:

$$p = 10^{-\frac{dec(ascii)}{10}}$$

ASCII TABLE



000		(nul)	016	►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺	(soh)	017	◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☻	(stx)	018	↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥	(etx)	019	‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦	(eot)	020	¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	♣	(enq)	021	§	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠	(ack)	022	▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	•	(bel)	023	↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	◘	(bs)	024	↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009		(tab)	025	↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010		(lf)	026		(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂	(vt)	027	←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	♀	(np)	028	∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013		(cr)	029	↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	♫	(so)	030	▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	☼	(si)	031	▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

In informatics every character has a number. For the computer they are actually interchangeable. The binary ′01000001′ refers at the same time to the number ′65′ and to the character ′A′. Only the context decides which definition will be used. It is therefore fairly straight forward to translate numbers between 0-127 to characters.

EXAMPLE FASTQ DECODING

```
@read1
TTACGTTTTT
+read1
87ba777777
```

Base 'A' in Illumina encoding (*offset* $= 64$)

- A has the quality b
- b $\Rightarrow$ 98 (from ascii table)
- $98 - 64 = 34$; my base quality for A is therefore 34
- error probability $p = 10^{-34/10} = 0.000398$

This error probability can be interpreted as 1 in 2511 ($= 1/0.000398$) base pairs being wrongly sequenced (= sequencing error)

EXERCISES FASTQ DECODING

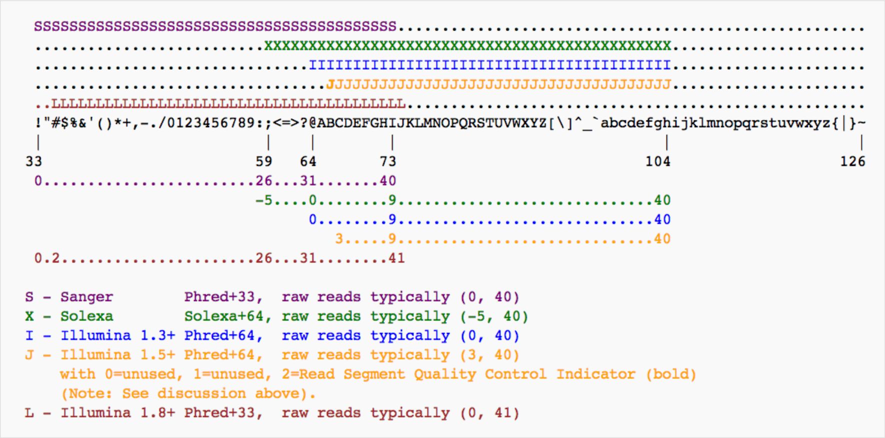
```
@read1
TTTACGTTTT
+read1
aaaa8aaaa
```

- Q1: Base quality of 'A' in Illumina
- Q2: Error probability of 'A' in Illumina
- Q3: Base quality of 'A' in Sanger
- Q4: Error probability of 'A' in Sanger
- Q5: Base quality of C in Sanger and Illumina. Is there anything weird about this results?
- Q6: Can you decide if the fastq file shown above is in Sanger or Illumina.

IS MY FASTQ-FILE SANGER OR ILLUMINA ENCODED?

If any of the bases has a negative quality with Illumina (64) than the encoding is Sanger (33). Therefore if you find any ascii character $<$ 64 (e.g.: 1,2,3,4,5,6,7,8,9) in your fastq file than your encoding is Sanger; if not than it is Illumina.
Exercise: which quality encoding is your fastq file?

Overview fastq



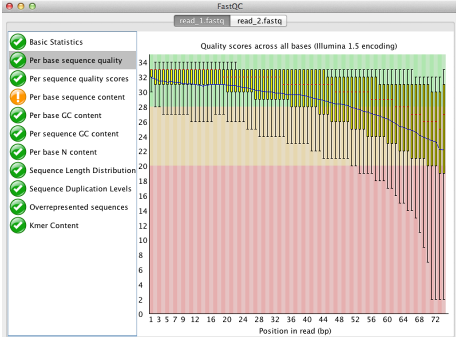
```
  SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
  ..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX......................
  ...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII......................
  .................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ......................
  ..LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL....................................................
  !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
  |                         |    |        |                              |                     |
 33                        59   64       73                            104                   126
  0........................26...31.......40
                           -5....0........9.............................40
                                 0........9.............................40
                                    3.....9.............................40
  0.2......................26...31........41

 S - Sanger        Phred+33,  raw reads typically (0, 40)
 X - Solexa        Solexa+64, raw reads typically (-5, 40)
 I - Illumina 1.3+ Phred+64,  raw reads typically (0, 40)
 J - Illumina 1.5+ Phred+64,  raw reads typically (3, 40)
     with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
     (Note: See discussion above).
 L - Illumina 1.8+ Phred+33,  raw reads typically (0, 41)
```

Note: in PoPoolation1/2 we call the offset of 33 sanger and the offset of 64 illumina, which is not entirely in agreement with the definition shown here (source Wikipedia). So, in theory, there are actually five fastq files, in practice however only the offset is relevant (33, 64).

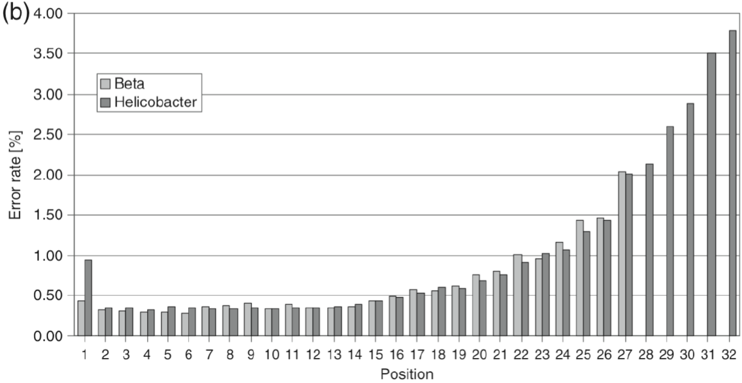
FastQC: Quality control of short reads

After obtaining the short read files, the first questions you are usually asking is: How successful has the sequencing of my reads been? What is the quality of my reads? The user-friendly tool FastQC can help to answer these questions (In Home/programs/FastQC double click fastqc).


FastQC
read_1.fastq
read_2.fastq
✓ Basic Statistics
✓ Per base sequence quality
✓ Per sequence quality scores
! Per base sequence content
✓ Per base GC content
✓ Per sequence GC content
✓ Per base N content
✓ Sequence Length Distribution
✓ Sequence Duplication Levels
✓ Overrepresented sequences
✓ Kmer Content
Quality scores across all bases (Illumina 1.5 encoding)
34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0
1 3 5 7 9 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72
Position in read (bp)

TRIMMING OF READS; WHY?

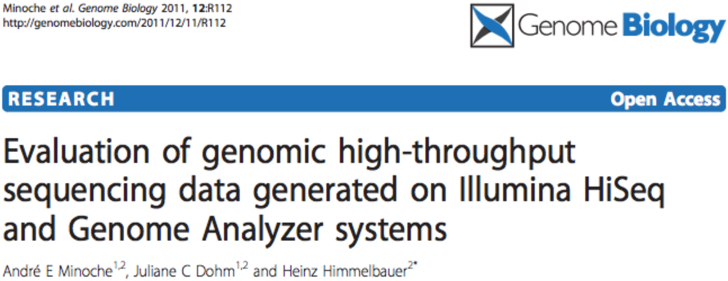
Before we map the paired end reads we need to deal with a problem: Error rate of the reads is increasing with the length. Also remember your FastQC results, the base quality is decreasing with the length of the reads.



Source: Dohm J. (2008) Substan)al biases in ultrashort read data sets from

Trimming of reads; Why?

A recent study clearly showed that trimming of the reads at low quality is the single quality filtering step that most dramatically improves the results (reduces analysis artefacts).



Minoche *et al. Genome Biology* 2011, **12**:R112
http://genomebiology.com/2011/12/11/R112

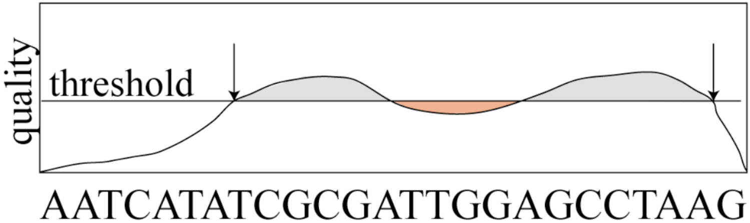
Genome **Biology**

RESEARCH **Open Access**

Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and Genome Analyzer systems

André E Minoche[1,2], Juliane C Dohm[1,2] and Heinz Himmelbauer[2*]

TRIMMING ALGORITHM OF POPOOLATION



- Given some arbitrary quality threshold (usually base quality of 20) the algorithm finds the highest scoring substring of the read.
- some fraction of the bases may be below the quality threshold, as long as a new high score can be achieved.
- the algorithm is very similar to dynamic programming (Smith-Waterman)
- handles single end reads as well as paired end reads

Trimming

```
mkdir peanalysis

# Trimming
perl ~/programs/popoolation/basic-pipeline/trim-fastq.pl --
    disable-zipped-output --input1 read_1.fastq --input2 read_2.
    fastq --min-length 50 --no-5p-trim --quality-threshold 20 --
    fastq-type illumina --output peanalysis/read_tr
```

- –input the input files
- –output the output prefix; this will create three files with the extensions _1 _2 _SE;
- –min-length discard reads that are after trimming smaller than this threshold; Note this step may create orphan reads, i.e.: reads who lost their mate :(
- –no-5p-trim only trim reads at the 3' end; this is necessary for the removal of duplicates
- –quality-threshold reads should on average have a score higher than this threshold
- –fastq-type is the encoding of the base quality in sanger or illumina (remember offset)
- –disable-zipped-output in the newest versions of PoPoolation the output of the fastq files is per default zipped. Here we disable this feature

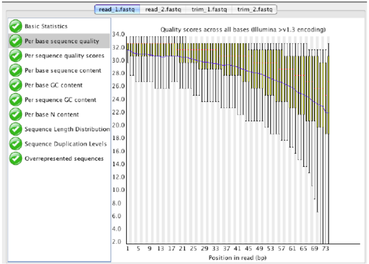
TRIM STATISTIC

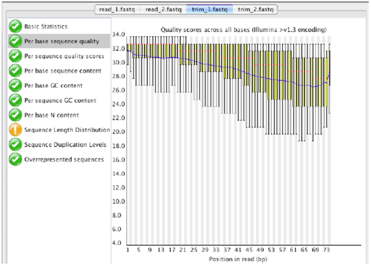
```
Read-pairs processed: 52322
Read-pairs trimmed in pairs: 52322
Read-pairs trimmed as singles: 0

FIRST READ STATISTICS
First reads passing: 52322
5p poly-N sequences trimmed: 0
3p poly-N sequences trimmed: 124
Reads discarded during 'remaining N filtering': 0
Reads discarded during length filtering: 0
Count sequences trimed during quality filtering: 19928

Read length distribution first read
length count
50 322
51 327
52 351
53 359
54 358
55 381
56 366
```

EFFECT OF TRIMMING ON QUALITY





Exercise:

- open FastQC
- load read_1.fastq
- load read_tr_1
- compare the base quality between trimmed and untrimmed
- compare the sequence length distribution between trimmed and untrimmed

BEFORE MAPPING, PREPARE THE REFERENCE GENOME (FASTA-FILE)

Now in addition to the fastq file we are encountering the fasta-file. To view the reference genome type:

```
less dmel-2R-chromosome-r5.22.fasta
```

and you obtain:

```
>2R type=chromosome_arm; loc=2R:1..21146708; ID=2R; dbxref=GB:AE013599; MD5=1589a9447d4dc94c048aa48ea5b8099d; length=21146708; release=r5.22; species=Dmel;
GACCCGCTAGGAGATGTTGAGATTGTGAGTACTTCTTGGAATTTGGTTAT
CTATTATAAAATGGATCCATATTTTAAAATGTTAACAAAGGGTAATGCGC
TTATACAAAGTATGAGGAAAGTTTGCGAAAGACTTCATAGCTTTGAAGAG
....
```

- symbol indicating start of a new sequence '>' directly followed, without space, by the name of the sequence; after space some description of the sequence
- the actual sequence, mostly in chunks of 60 nucleotides per line (sometimes 50, the length is irrelevant)
- optional: additional sequences with the same formatting

Preparing the reference sequence for mapping

```
mkdir peanalysis/wg
awk ’{print $1}’ dmel-2R-chromosome-r5.22.fasta
    > peanalysis/wg/dmel-2R-short.fasta
bwa index peanalysis/wg/dmel-2R-short.fasta
```

Note; the awk command just prints the first column, where a column is defined as everything being separated by a space. With this command we are therefore removing the description of the fasta entry. This step is strongly recommended as unnecessarily long fasta identifiers may lead to problems in downstream analysis.

```
>2R type=chromosome_arm; loc=2R:1..21146708; ID=2R; dbxref=GB:AE013599; MD5=1589
    a9447d4dc94c048aa48ea5b8099d; length=21146708; release=r5.22; species=Dmel;
GACCCGCTAGGAGATGTTGAGATTGTGAGTACTTCTTGGAATTTGGTTAT
...

=> transformed into:

>2R
GACCCGCTAGGAGATGTTGAGATTGTGAGTACTTCTTGGAATTTGGTTAT
...
```

Mapping software

- BWA ⇐
- Bowtie
- GEM
- BFAST
- STAMPY
- MAQ
- SOAP
- BLAT

In this introduction, we are using BWA for aligning short reads to the reference genome. BWA is currently one of the most widely used alignment algorithm. It is fast and flexible (many options)

Paired end mapping

```
bwa aln -I -m 100000 -o 1 -n 0.01 -l 200 -e 12 -d 12 -t 2
    peanalysis/wg/dmel-2R-short.fasta peanalysis/read_tr_1 >
    peanalysis/read_tr_1.sai
bwa aln -I -m 100000 -o 1 -n 0.01 -l 200 -e 12 -d 12 -t 2
    peanalysis/wg/dmel-2R-short.fasta peanalysis/read_tr_2 >
    peanalysis/read_tr_2.sai
```

- -I input is in Illumina encoding (offset 64); do not provide this when input is in sanger! Very important parameter!
- -m not important; just telling bwa to process smaller amounts of reads at once
- -l 200 seed size (needs to be longer than the read length to disable seeding)
- -e 12 -d 12 gap length (for insertions and deletions)
- -o 1 maximum number of gaps
- -n 0.01 the number of allowed mismatches, in terms of probability. In general the lower the value the more mismatches are allowed. The exact translation is shown at the beginning of the mapping
- -t 2 number of threads, the more the faster

CREATING PAIRED END INFORMATION

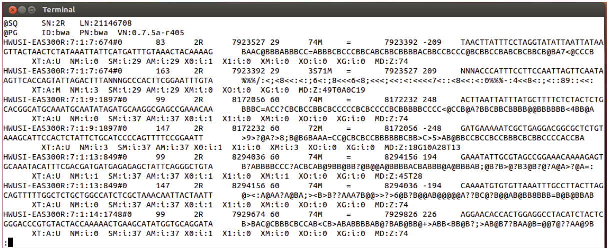
The reads are actually mapped as single ends. Only the 'bwa sampe' step creates the paired end information.

```
bwa sampe peanalysis/wg/dmel-2R-short.fasta
    peanalysis/read_tr_1.sai peanalysis/
    read_tr_2.sai peanalysis/read_tr_1
    peanalysis/read_tr_2 > peanalysis/pe.sam
```

INSPECTING THE SAM-FILE

```
less peanalysis/pe.sam
```

and you will get something like



```
@SQ     SN:2R   LN:21146708
@PG     ID:bwa  PN:bwa  VN:0.7.5a-r405
HWUSI-EAS300R:7:1:7:674#0       83      2R      7923527 29      74M     =       7923392 -209    TAACTTATTTCCTAGGTATATTAATTATAA
GTTACTAACTCTATAAATTATTCATGATTTGTAAACTACAAAAG      BAAC@BBBABBBCC=ABBBCBCCCBBCABCBBCBBBBACBBCCBCCC@BCBBCCBABCBCBBCB@BA7<@CCCB
      XT:A:U  NM:i:0  SM:i:29 AM:i:29 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:74
HWUSI-EAS300R:7:1:7:674#0       163     2R      7923392 29      3S71M   =       7923527 209     NNNACCCATTTCCTTCCAATTAGTTCAATA
AGTTCACCAGTATTAGACTTTANNNGCCCACTTCGGAATTTGTA      %%%/:<;<8<<:<:;6<:;8<<<6<8;<<<;<<:<:<<<<7<::<8<<:<:0%%%-:4<<8<:;<::89::<<:
      XT:A:M  NM:i:3  SM:i:29 AM:i:29 XM:i:0  XO:i:0  XG:i:0  MD:Z:49T0A0C19
HWUSI-EAS300R:7:1:9:1897#0      99      2R      8172056 60      74M     =       8172232 248     ACTTAATTATTTATGCTTTTCTCTACTCTG
CACGGCATGCAAATGCAATATAGATGCAAGGCGAGCCGAAACAA      BBBC=ACC?CBCBCCBBCBCCCCCBCBCCCCBCBBBBBCCCC<@CCB@A?BBCBBCBBBB@@BBBBBB<4BB@A
      XT:A:U  NM:i:0  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:74
HWUSI-EAS300R:7:1:9:1897#0      147     2R      8172232 60      72M     =       8172056 -248    GATGAAAAATCGCTGAGGACGGCGCTCTGT
AAAGCATTCCACTCTATTCTGCATCCCCAGTTTTCCGGAATT        >9>?@A?>8;B@B6BAAA=CC@CBCBCCBBBBBBCBB>C>5>AB@BBCCBCCBCCBBBCBCBBCCCCACCBA
        XT:A:U  NM:i:3  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:3  XO:i:0  XG:i:0  MD:Z:18G10A28T13
HWUSI-EAS300R:7:1:13:849#0      99      2R      8294036 60      74M     =       8294156 194     GAAATATTGCGTAGCCGGAAACAAAAGAGT
GCAAATACATTTCGACGATGATGAGAGAGCTATTCAGGGCTGTA      B?ABBBBCCC?ACBCAB@9BB@BB?@B@@A@BBBBACBABBB@A@BBBAB;@B?B>@?B3@B?@?A@A>?@A=:
      XT:A:U  NM:i:1  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0  MD:Z:45T28
HWUSI-EAS300R:7:1:13:849#0      147     2R      8294156 60      74M     =       8294036 -194    CAAAATGTGTGTTAAATTTGCCTTACTTAG
CAGTTTTTGGCTCTGCTGGCCATCTCGCTAAACAATTACTAATT      @><:A@AA?A@BA;><B>B??AAA7B@@>>?>6@B?B@@AB@@@@@A??BC@?B@@AB@BB8BBB=B@B@BBAB
      XT:A:U  NM:i:0  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:74
HWUSI-EAS300R:7:1:14:1748#0     99      2R      7929674 60      74M     =       7929826 226     AGGAACACCACTGGAGGCCTACATCTACTC
GGGACCCGTGTACTACCAAAAACTGAAGCATATGGTGCAGGATA      B>BAC@CBBBCBCCAB<CB>ABABBBBAB@?BAB@BB@+>ABB<BB@B?;>AB@B7?BAA@B=@@7@??AA@9B
      XT:A:U  NM:i:0  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:74
:
```

- samtools: `http://samtools.sourceforge.net/`
- documentation: `http://samtools.sourceforge.net/SAMv1.pdf`

SAM FILE

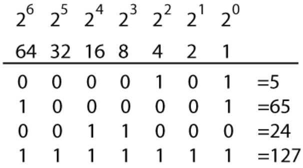
A sam file has two parts, first a general info (lines starting with an '@') and second the aligned reads (all other lines) which have the following columns

- col 1: name of the read
- col 2: binary flag (contains lots of useful information, compressed into one number...)
- col 3: ID of the reference chromosome (e.g.: a read may align to 2L or 3R ...)
- col 4: position of the read in the reference chromosome (most 5′ position is given)
- col 5: mapping quality (do not mix up with base quality)
- col 6: CIGAR string; specifies the alignment of the read with the reference chromosome
- col 10: sequence of the read (like in fastq)
- col 11: base quality of read (like in fastq); Per definition, the quality encoding has to be Sanger! (There should be only one version of sam file as opposed to the 5 fastq files)
- col 12+: optional information; dependent on the mapping software

SAM FILE; COLUMN 2 - BINARY FLAG

They developed a human readable format (sam) and a computer readable format (binary sam), yet they included a binary flag into the human readable format...seriously...wtf..

However, as the sam file is a currently quite standard we still have to learn the meaning of this flag. What you actually find in column 2 is an integer. In informatics every integer can be displayed as binary number:



2^6	2^5	2^4	2^3	2^2	2^1	2^0	
64	32	16	8	4	2	1	
0	0	0	0	1	0	1	=5
1	0	0	0	0	0	1	=65
0	0	1	1	0	0	0	=24
1	1	1	1	1	1	1	=127

In the binary flag of the sam-file, every of these 0 or 1 has a meaning. For example a 1 at the third position (**from the right**) means that the read could not be mapped to the reference genome (e.g.: to many mismatches) whereas a 0 at the third position means the read could be mapped to the reference genome

THE MOST IMPORTANT FLAGS

To make matters worse, the actual meaning of every flag is provided in hexadecimal numbers. Overview of the most important flags. For more details see `http://samtools.sourceforge.net/SAMv1.pdf`.

- 0x1 = first position from the right; if set to 1, than the read is a part of a paired end read (0 means single end)
- 0x2 = second position: if set to 1, than the reads map as a proper pair (definition depends on mapping software)
- 0x4 = third position: the read has not been mapped (remember 0 means mapped)
- 0x8 = fourth position: the mate (the other read of the pair) could not be mapped
- 0x10 = fifth position: the read maps as reverse complement (the sequence in col10 has been reverse complemented to)
- 0x20 = sixth position: the mate maps as reverse complement
- 0x100 = ninth position: this is a secondary alignment (thus there is an alternative more suitable alignment for the given read)
- 0x200 = tenth position: the read does not pass quality control (Illuminas requirements)
- 0x400 = eleventh position: the read is a duplicate (either PCR or optical); a suitable software has to be used to identify duplicates (e.g.: Picard)

QUICK WAY TO VISUALIZE BINARY FLAGS?

```
echo ’83’ |wcalc -b
echo ’0x20’|wcalc -b
```

A binary flag (e.g.: 0x20) is set if ’col 2’ (e.g.: 83) contains a bit (1) at the given position. In the following example the flag is not set

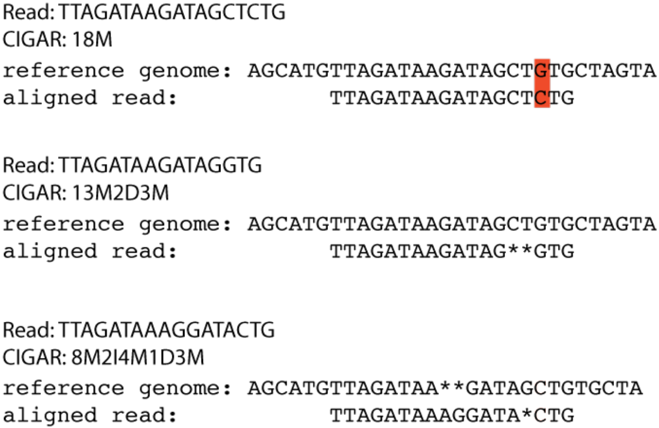
```
0b1010011 (83)
 0b100000 (0x20)
```

In the following example the flag is set

```
0b1010011 (83)
     0b10 (0x2)
```

CIGAR STRING (COL 6)

The CIGAR string ('col 6') only specifies the alignment between the read and the reference genome. Remember the starting position is given in 'col 4' and the sequence of the read in 'col 10'. Note mismatches are not affecting the CIGAR string



```
Read: TTAGATAAGATAGCTCTG
CIGAR: 18M
reference genome: AGCATGTTAGATAAGATAGCTGTGCTAGTA
aligned read:           TTAGATAAGATAGCTCTG


Read: TTAGATAAGATAGGTG
CIGAR: 13M2D3M
reference genome: AGCATGTTAGATAAGATAGCTGTGCTAGTA
aligned read:           TTAGATAAGATAG**GTG


Read: TTAGATAAAGGATACTG
CIGAR: 8M2I4M1D3M
reference genome: AGCATGTTAGATAA**GATAGCTGTGCTA
aligned read:           TTAGATAAAGGATA*CTG
```

CONVERTING SAM TO BAM

- sam.. Sequence Alignment Map format ⇒ optimized for humans
- bam.. binary sam ⇒ optimized for computers

It is easily possible to convert a sam to bam and vice versa a bam to sam. In the following we convert a sam into a bam and finally sort the bam file

```
samtools view -Sb peanalysis/pe.sam > peanalysis/pe.
    bam
```

- -S input is sam
- -b output is bam (-S may be merged with -b to -Sb)
- 'sort - outpufile' input for sorting is the pipe (rather than a file)

SORTING WITH PICARD

Here we use Picard to sort reads, which is a bit more complicated than sorting with samtools. Sorting with Picard is however necessary as otherwise the downstream analysis (MarkDuplicates) would not work.

```
java -Xmx2g -jar ~pic/SortSam.jar I=peanalysis/pe.sam O=
     peanalysis/pe.sort.sam VALIDATION_STRINGENCY=SILENT SO=
     coordinate
```

- Picard runs with Java
- -Xmx2g give Java 2 Gb of memory
- -jar SortSam use the Java software SortSam
- I= input
- O= output
- SO= sort order; sort by coordinate
- VALIDATION_STRINGENCY= Picard is like a Princess that is constantly complaining about every small deviation of our sam file from the most stringent requirements. I have never found a sam file satisfying all of Picards demands ⇒ 'shut up Picard'

LET’S HAVE A FIRST VIEW OF THE MAPPING RESULTS

First we need to index the bam file (necessary for visualization but nothing else)

```
samtools index peanalysis/pe.sort.bam
mkdir peanalysis/igv
java -Xmx2g -jar ~/programs/IGV_2.3.26/igv.jar
```

- Genomes ⇒ Create .genome File; Unique identifier=”Dmel2R”; Desriptive name=”Drosophila chromosome 2R”; FASTA file=’peanalysis/wg/dmel-2R-short.fasta’;
- Save genome as ’peanalysis/igv/Dmel2R.genome’
- Load the mapped reads (peanalysis/pe.sort.bam)
- Zoom in at position 8,250,000
- Inspect the alignments

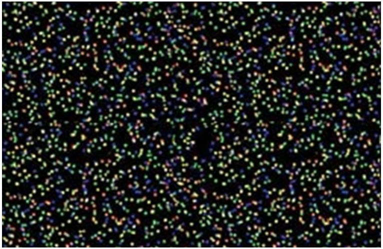
Pay special attention to

- the coverage graph on top
- SNPs (colored bases in the alignment); zoom in and out
- reference sequence (zoom in and out)
- alignment strands of the reads (find some forward and some reverse aligned reads);
- right click; color alignments by read strand
- find some insertions and deletions
- hover over read, what is happening now?
- Right click on alignment ⇒ View as pairs
- Find some orphan reads (reads without mate). What could be the reason for orphan reads?

DUPLICATES

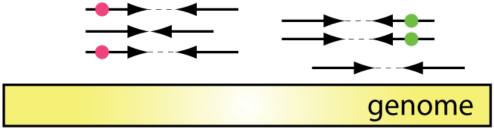
There are two sources of duplicates with the Illumina technology

- Optical duplicates: they are occurring during the sequencing step; The algorithm responsible for identifying isolated clusters wrongly identifies a single cluster as two (see picture below).
- PCR duplicates; occurs during sample preparation where a PCR step is necessary to amplify the amount of DNA for the sequencing.



How to recognize duplicates

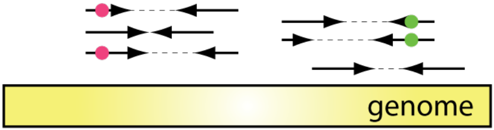
One common proxy is to use the mapping positions of PE reads to recognize duplicates where reads having exactly identical positions are marked as duplicates. This has the advantage that it also allows for sequencing errors within duplicated reads. The chances that two PE reads accidentally have identical positions are minimal.



Duplicates are marked by dots having identical colors.

Duplicates with trimmed reads

During trimming reads may be truncated at sequences having a low quality, thus duplicated reads may end up having different lengths. Fortunately, removal of duplicates can still be performed if only the 3′ ends of reads are trimmed (remember that the quality deteriorates mostly at the 3′ end of reads). This is because Picard recognizes duplicates by PE reads having identical 5′ positions (5′ of the read not the genome).



Duplicates are marked by dots having identical colors.

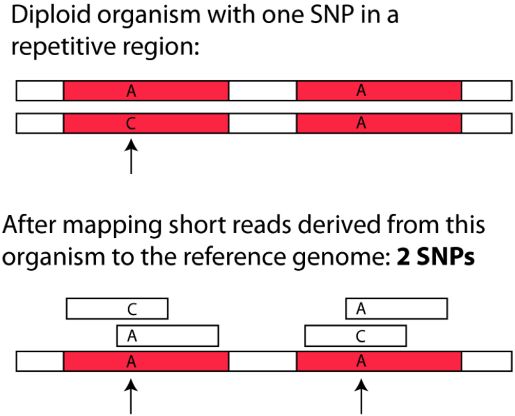
REMOVING DUPLICATES

```
# the following only accepts a sam file sorted by Picard.
java -Xmx2g -jar ~pic/MarkDuplicates.jar I=peanalysis/pe.sort.sam O=peanalysis/pe.rmd.sort.sam M=peanalysis/dupstat.txt VALIDATION_STRINGENCY=SILENT REMOVE_DUPLICATES=true
```

- I= input file
- O= output file for reads
- M= output file of statistics (how many identified duplicates)
- REMOVE_DUPLICATES= remove duplicates from the output file rather than just marking them (remember flag in sam-file 0x400)

Another problem, ambiguous mapping

Ambiguously mapped reads can lead to wrong SNPs.


Diploid organism with one SNP in a repetitive region:
A
A
C
A
After mapping short reads derived from this organism to the reference genome: **2 SNPs**
C
A
A
C
A
A

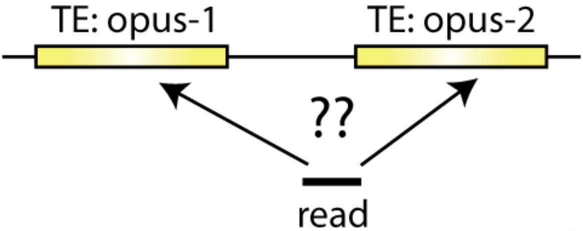
⇒ therefore exclude ambiguously mapped reads

AMBIGUOUS MAPPING POSITION AND MAPPING QUALITY

Remember column 5 of the sam file contains the mapping quality. Similarly to the base quality, the mapping quality is the log scaled probability that the position of the read is wrong.

- 20.. one out of 100 reads is wrongly mapped
- 30.. one out of 1000 reads is wrongly mapped
- 0.. every read is wrongly mapped

A major cause for incorrect or ambiguous mapping positions are repetitive regions in the genome



REMOVE LOW QUALITY ALIGNMENTS

The following command ensures that we remove ambiguously mapped reads and only retain PE reads where both mates align with the reference genome

```
samtools view -q 20 -f 0x0002 -F 0x0004 -F 0x0008 -b
     peanalysis/pe.rmd.sort.bam > peanalysis/pe.q20.
    rmd.sort.bam
```

- -q 20 only keep reads with a mapping quality higher than 20 (remove ambiguously aligned reads)
- -f 0x0002 only keep proper pairs (remember flags from sam file)
- -F 0x0004 remove reads that are not mapped
- -F 0x0008 remove reads with an un-mapped mate

Note '-f' means only keep reads having the given flag and '-F' discard all reads having the given flag.

COMPARE FILTERED WITH UNFILTERED READS

```
samtools index peanalysis/pe.q20.rmd.sort.bam
java -Xmx2g -jar ~/programs/IGV_2.3.26/igv.jar
```

- Load the unfiltered reads (peanalysis/pe.sort.bam)
- Load the filtered reads (peanalysis/pe.q20.rmd.sort.bam)
- Zoom in at position 8,250,000
- Compare the two files; Can you identify any duplicates that have been removed?
- Compare the two files; Can you find any filtered ambiguous reads?

2R:8,172,161

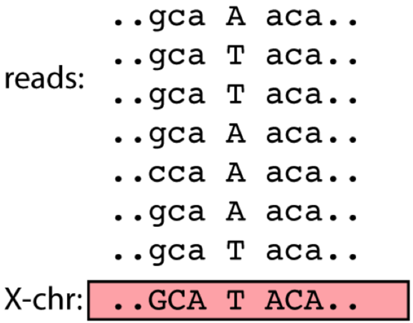
2R:8,113,549

CREATING A MPILEUP FILE

```
cd peanalysis
samtools mpileup -B -Q 0 -f wg/dmel-2R-short.
    fasta pe.q20.rmd.sort.bam > pe.mpileup
less pe.mpileup
```

- -B disable BAQ computation (base alignment quality)
- -Q skip bases with base quality smaller than the given value
- -f path to reference sequence

WHAT IS A PILEUP?



```
        ..gca A aca..
        ..gca T aca..
reads:  ..gca T aca..
        ..gca A aca..
        ..cca A aca..
        ..gca A aca..
        ..gca T aca..
X-chr:  ..GCA T ACA..
```

resulting pileup entry:

```
X-chr  2312  T   7   A..AAA.   SUUTTBB
```

MPILEUP FILE

```
2R 7809811 C 18 .............,..,. B=9>=C<BBB<@A4BC6C
2R 7809812 A 18 .............,..,. @B;66:7ABB@7A8CB;B
2R 7809813 G 18 .............,..,. AA9/:C<<?B@@B?BBAC
```

- col1 reference chromosome
- col2 position
- col3 reference character
- col4 coverage
- col5 bases for the given position ('.' identical to reference character - forward strand; ',' identical to reference - reverse strand)
- col6 quality for the bases

Filtering indels

```
perl ~/programs/popoolation/basic-pipeline/identify-
    genomic-indel-regions.pl --indel-window 5 --min-
    count 2 --input pe.mpileup --output indels.gtf
```

- –indel-window how many bases surrounding indels should be ignored
- –min-count minimum count for calling an indel. Note that indels may be sequencing errors as well

```
perl ~/programs/popoolation/basic-pipeline/filter-
    pileup-by-gtf.pl --input pe.mpileup --gtf indels
    .gtf --output pe.idf.mpileup
```

Note: the filter-pileup script could also be used to remove entries overlapping with transposable elements (RepeatMasker produces a gtf as well).

SUBSAMPLING TO UNIFORM COVERAGE

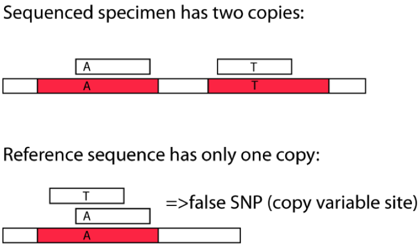
Several population genetic estimators are sensitive to sequencing errors. For example a very low Tajima's D, usually indicative of a selective sweep, may be, as an artifact, frequently be found in highly covered regions because these regions have just more sequencing errors. To avoid these kinds of biases we recommend to subsample to an uniform coverage.

```
perl ~/programs/popoolation/basic-pipeline/subsample-pileup.pl --min-qual 20 --method withoutreplace --max-coverage 50 --fastq-type sanger --target-coverage 10 --input pe.idf.mpileup --output pe.ss10.idf.mpileup
```

- –min-qual minimum base quality
- –method method for subsampling, we recommend without replacement
- –target-coverage which coverage should the resulting mpileup file have
- –max-coverage the maximum allowed coverage, regions having higher coverages will be ignored (they may be copy number variations and lead to wrong SNPs)
- –fastq-type (sanger means offset 33)

Copy number variations may lead to wrong SNPs

We introduced a maximum coverage because copy number variations could lead to wrong SNPs.



Sequenced specimen has two copies:

Reference sequence has only one copy:

=>false SNP (copy variable site)

The signatures of these artefactual SNPs are:

- fairly balanced allele frequencies (e.g.: 50% A and 50% T)
- high coverage ← targeted by PoPoolation

INSPECTING THE SUB-SAMPLED PILEUP

```
less pe.ss10.idf.mpileup
```

```
2R 7799887 A 10 AAAAAAAAAA 5555555555
2R 7799889 G 10 GGGGGGGGGG 5555555555
2R 7799890 G 10 GGGGGGGGGG 5555555555
2R 7799892 C 10 CCCCCCCCCC 5555555555
2R 7799893 A 10 AAAAAAAAAA 5555555555
```

Note that the quality has been uniformly set to the ’–min-qual’

Bioinformatics workflow: Finally PoPoolation

After these many steps we can finally proceed and estimate the polymorphism in the population with PoPoolation. Now it's also time to apologize for the length of this pipeline. When we started with PoPoolation this pipeline was much shorter. But gradually we eliminated possible confounding factors (e.g.: duplicates, subsampling) and the pipeline grew. However, I wanted to give you the full picture, so that you are immediately able to produce high quality results.

- trimming of reads
- mapping of reads
- removing duplicates
- removing low quality sequences (ambiguous mapping)
- converting to mpileup
- subsampling to uniform coverage
- remove regions in the neighboring to indels
- PoPoolation ←

GETTING HELP

You can get help for any PoPoolation script with the option '–help'.

```
perl ~/programs/popoolation/Variance-sliding.pl
    --help
```

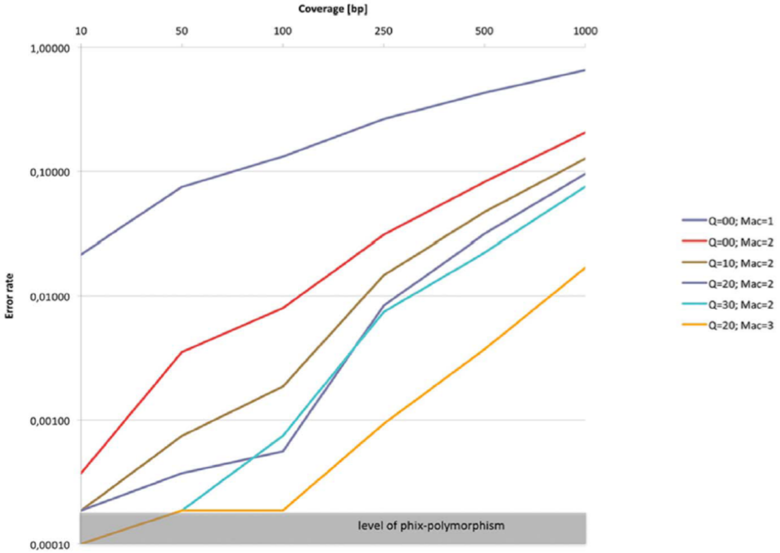
Exit help with pressing 'q'

Calculating Tajima's π

```
perl ~/programs/popoolation/Variance-sliding.pl --fastq-type
    sanger --measure pi --input pe.ss10.idf.mpileup --min-count
    2 --min-coverage 4 --max-coverage 10 --min-covered-fraction
    0.5 --pool-size 500 --window-size 1000 --step-size 1000 --
    region 2R:7800000-8300000 --output cyp6gi.pi --snp-output
    cyp6gi.snps
```

- –min-coverage –max-coverage: for subsampled files not important; should contain target coverage, i.e.: 10
- –min-covered-fraction minimum percentage of sites having sufficient coverage in the given window
- –min-count minimum occurrence of allele for calling a SNP
- –measure which population genetics measure should be computed (pi/theta/D)
- –pool-size number of chromosomes (thus number of diploids times two)
- –region compute the measure only for a small region; default is the whole genome
- –output a file containing the measure (π) for the windows
- –snp-output a file containing for every window the SNPs that have been used for computing the measure (e.g. π)
- –window-size –step-size control behavior of sliding window; if step size is smaller than window size than the windows will be overlapping.

How to choose a minimum count threshold?



Output

```
less cyp6gi.pi
```

```
2R 7800500 0 0.218 na
2R 7801500 6 0.683 0.004936240
2R 7802500 13 0.916 0.008076347
2R 7803500 3 0.782 0.002411416
2R 7804500 6 0.599 0.006439348
```

- col 1: reference chromosome
- col 2: position of window (mean value)
- col 3: number of SNPs in the given window
- col 4: fraction of sites in the window having sufficient coverage ($min \leq x \leq max$)
- col 5: measure for the window (π)

SNP Output

```
less cyp6gi.snps
```

The file contains the SNP found in each window

```
>2R:7801500 2R:7801000-7802000 snps:6
2R 7801059 T 10 2 8 0 0 0
2R 7801066 G 10 6 0 0 4 0
```

- col 1: reference chromosome
- col 2: position of SNP
- col 3: reference character
- col 4: coverage
- col 5-9: counts of A, T, C, G, N respectively

Visualize in IGV

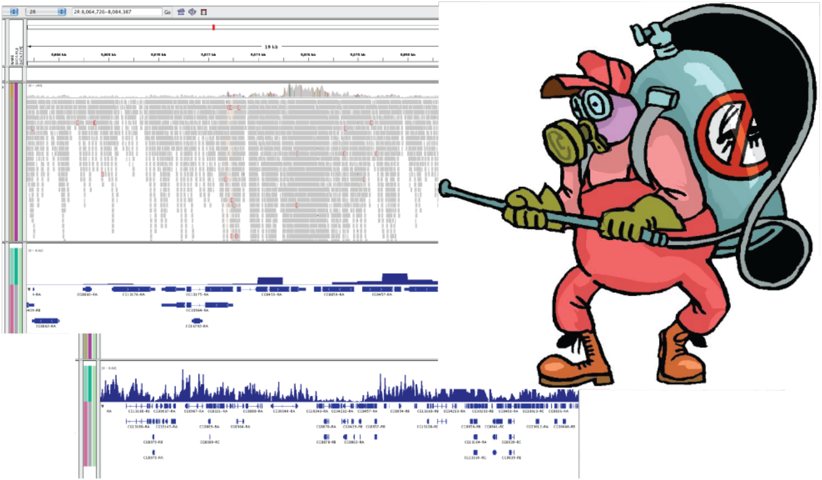
```
perl ~/programs/popoolation/VarSliding2Wiggle.pl --input cyp6g1.pi --trackname "pi" --output cyp6gi.wig
java -Xmx2g -jar ~/programs/IGV_2.3.26/igv.jar
```

Than:

- load the paired end reads (pe.q20.rmd.sort.bam)
- load the annotation (../cyp6g1.gtf)
- load the variation (cyp6g1.wig)
- search gene CG8453 (=cyp6g1)

Background: some variants around Cyp6g1 have recently swept to fixation in *D. melanogaster* as the gene confers resistance to DDT. So we would expect an extreme dip in variability in the neighborhood of this gene. However, the situation is complex as there has also been a lot of copy number variations (CNVs e.g.: duplications). Zoom in and inspect the coverage to see CNV regions.

Biology, here I come..



ADDITIONAL FEATURES

PoPoolation also allows to:

- Calculate Tajima's D, Wattersons Θ
- Calculate the measure (π, D, Θ) for genes (instead of windows)
- Calculate the measure for synonymous and non-synonymous sites
- Compute the divergence between two species using a Mauve alignment