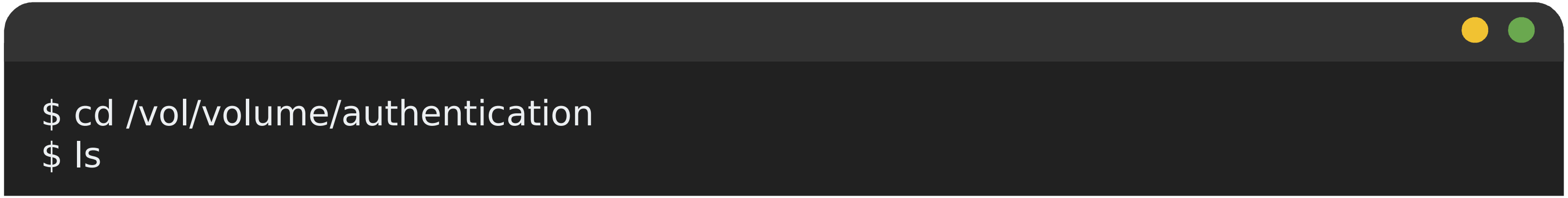
Lecture + Practical: General Authentication

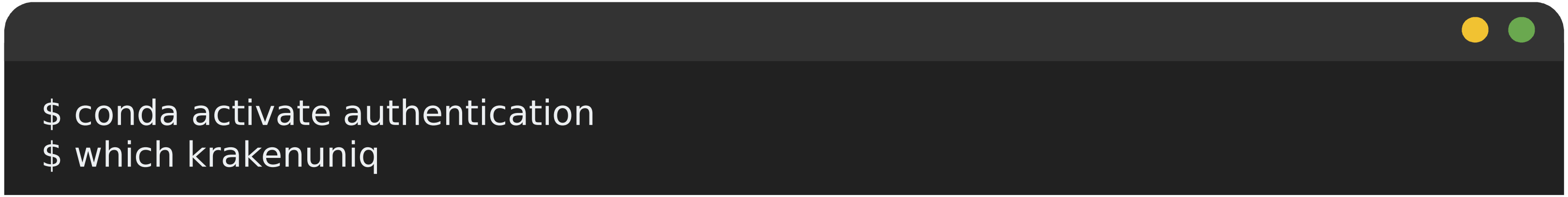
Nikolay Oskolkov

Before we start!



```
$ cd /vol/volume/authentication
$ ls
```

You should see at least a file called <session_name>.yaml



```
$ conda activate authentication
$ which krakenuniq
```

Should see a path with miniconda/ in it

Outline

- Genomic hit confirmation (how we see a true-positive taxonomic hit)
    - Modern validation criteria
        - evenness and breadth of coverage
        - alignment quality (edit distance, mapq)
        - affinity to reference (percent identity, multi-allelic SNPs)
    - Ancient-specific validation
        - deamination profile (PMD scores)
        - DNA fragmentation

Genomic Hit Confirmation: modern and ancient-specific validation criteria

Ways to detect ancient organisms

1) Alignment:

BOW TIE

BWA
stands for
Burrows Wheeler Aligner
Abbreviations.com

HOPS

2) Classification:

KRAKEN

Centrifuge

MetaPhlan

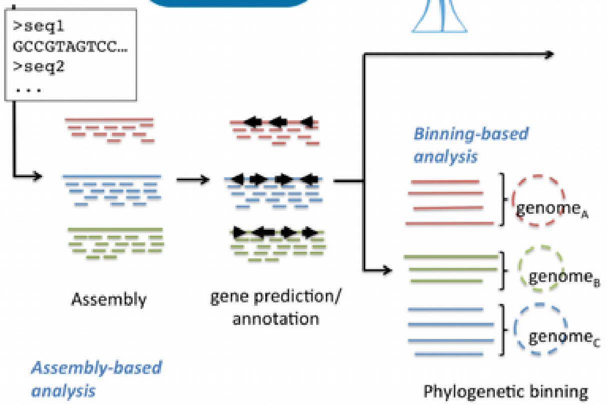
Clark

Reference based:
assume similarity to reference

3) De-novo assembly:

MEGA HIT

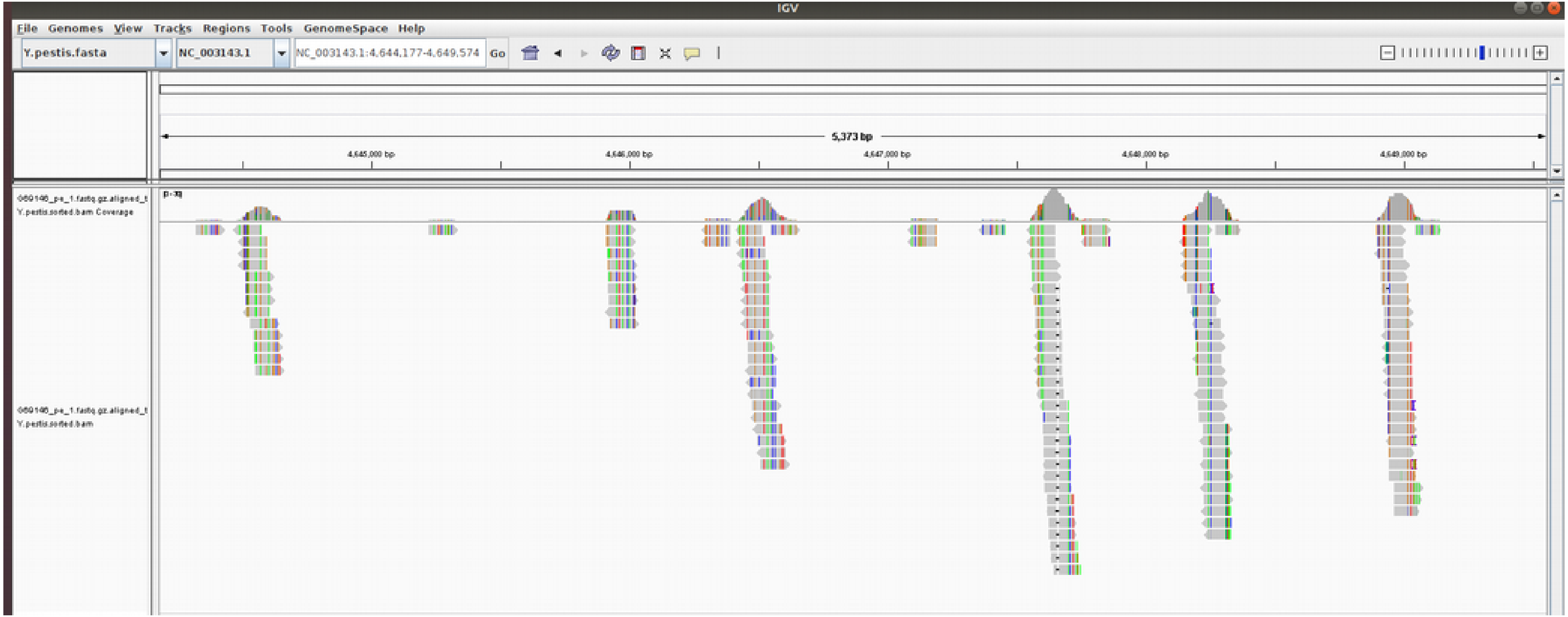
meta



```
>seq1
GCCGTAGTCC…
>seq2
...
```

Reference free:
rapidly developing but challenging

How to see a false-positive finding



Y.pestis.fasta

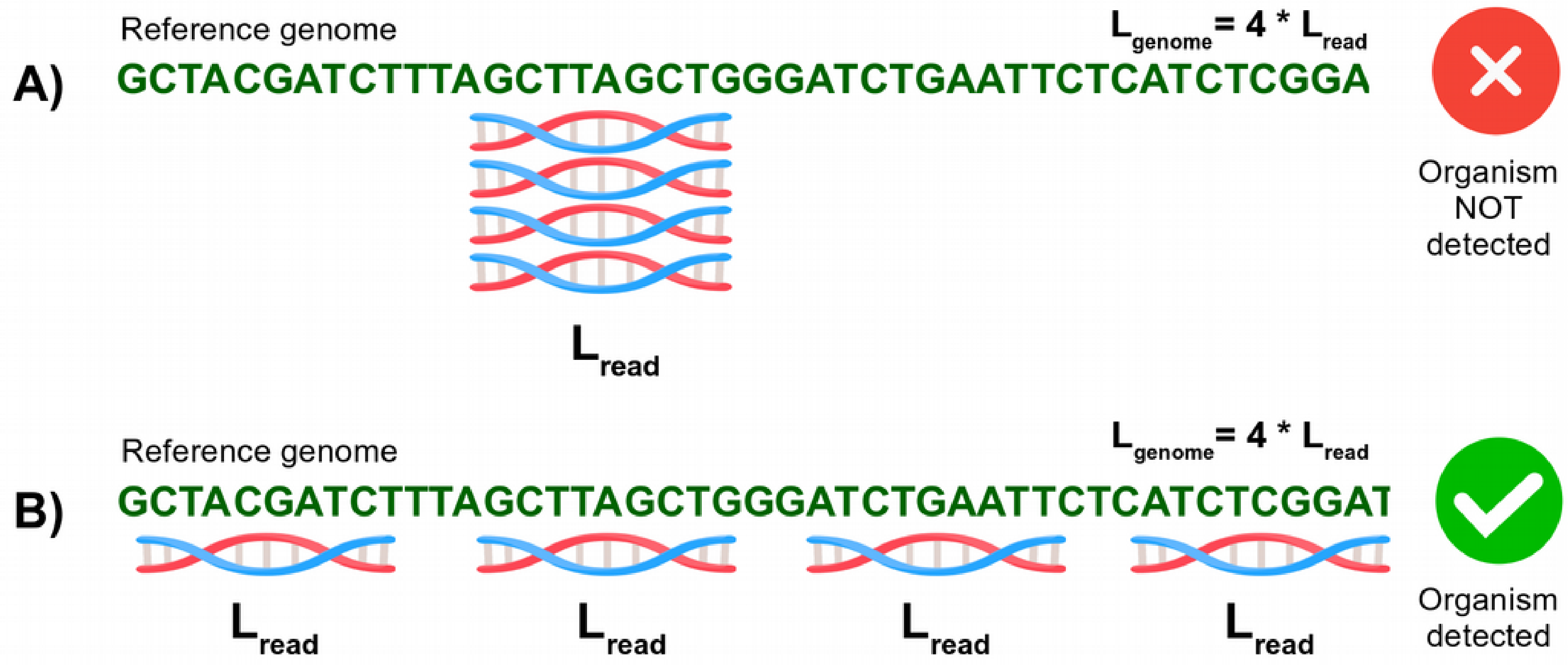
NC_003143.1

NC_003143.1:4,644,177-4,649,574 Go

IGV
File Genomes View Tracks Regions Tools GenomeSpace Help
5,373 bp
4,645,000 bp
4,646,000 bp
4,647,000 bp
4,648,000 bp
4,649,000 bp
[0-39]
060146_pe_1.fastq.gz.aligned_t
Y.pestis.sorted.bam Coverage
060146_pe_1.fastq.gz.aligned_t
Y.pestis.sorted.bam

~20 000 reads mapped uniquely to ***Yersinia pestis*** reference

Depth vs. breadth and evenness of coverage


A)
Reference genome
$L_{genome} = 4 * L_{read}$
GCTACGATCTTTAGCTTAGCTGGGATCTGAATTCTCATCTCGGA
L_{read}
Organism NOT detected
B)
Reference genome
$L_{genome} = 4 * L_{read}$
GCTACGATCTTTAGCTTAGCTGGGATCTGAATTCTCATCTCGGAT
L_{read}
L_{read}
L_{read}
L_{read}
Organism detected

Both A) and B) have identical depth of coverage:

Coverage = $(N_{reads} * L_{read}) / L_{genome} = (4 * L) / (4 * L) = 1X$

Let us compute breadth / evenness of coverage

We will use simulated ancient metagenomic data from Pochon et al. 2023

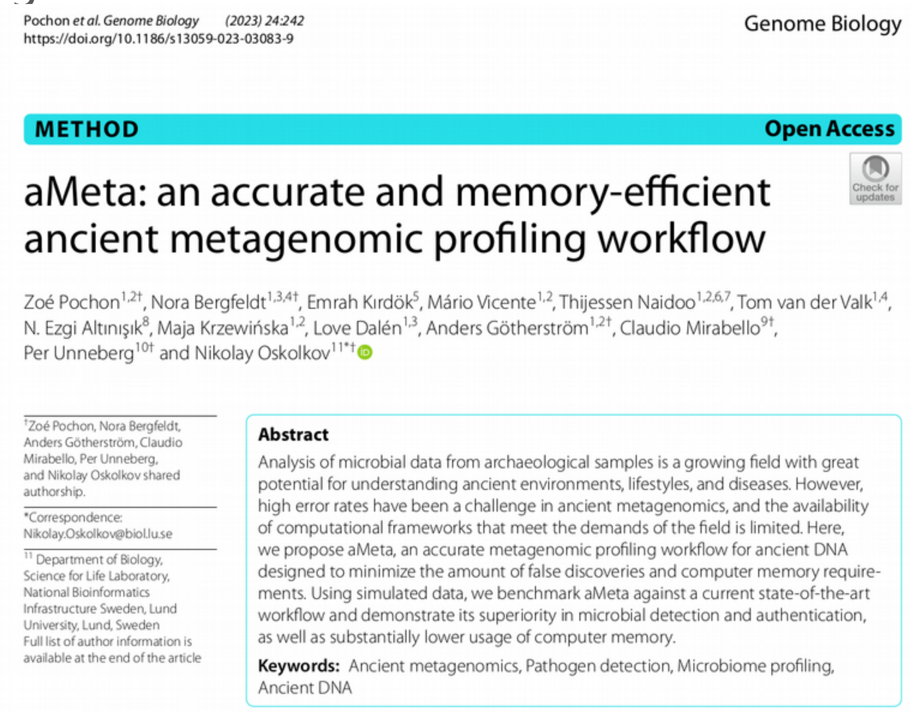
The simulated data can be accessed via:

https://doi.org/10.17044/scilifelab.21261405

We will profile the data with KrakenUniq

KrakenUniq database can be accessed via:

https://doi.org/10.17044/scilifelab.21299541



Pochon *et al. Genome Biology* (2023) 24:242
https://doi.org/10.1186/s13059-023-03083-9

Genome Biology

METHOD Open Access

aMeta: an accurate and memory-efficient ancient metagenomic profiling workflow

Zoé Pochon[1,2†], Nora Bergfeldt[1,3,4†], Emrah Kırdök[5], Mário Vicente[1,2], Thijessen Naidoo[1,2,6,7], Tom van der Valk[1,4], N. Ezgi Altınışık[8], Maja Krzewińska[1,2], Love Dalén[1,3], Anders Götherström[1,2†], Claudio Mirabello[9†], Per Unneberg[10†] and Nikolay Oskolkov[11*†]

†Zoé Pochon, Nora Bergfeldt, Anders Götherström, Claudio Mirabello, Per Unneberg, and Nikolay Oskolkov shared authorship.

*Correspondence: Nikolay.Oskolkov@biol.lu.se

[11] Department of Biology, Science for Life Laboratory, National Bioinformatics Infrastructure Sweden, Lund University, Lund, Sweden
Full list of author information is available at the end of the article

Abstract

Analysis of microbial data from archaeological samples is a growing field with great potential for understanding ancient environments, lifestyles, and diseases. However, high error rates have been a challenge in ancient metagenomics, and the availability of computational frameworks that meet the demands of the field is limited. Here, we propose aMeta, an accurate metagenomic profiling workflow for ancient DNA designed to minimize the amount of false discoveries and computer memory requirements. Using simulated data, we benchmark aMeta against a current state-of-the-art workflow and demonstrate its superiority in microbial detection and authentication, as well as substantially lower usage of computer memory.

Keywords: Ancient metagenomics, Pathogen detection, Microbiome profiling, Ancient DNA

Preprocess ancient metagenomic data

Download simulated ancient metagenomic reads:

```
wget https://figshare.scilifelab.se/ndownloader/articles/21261405/versions/1 \
&& export UNZIP_DISABLE_ZIPBOMB_DETECTION=true && unzip 1 && rm 1
```

Activate the conda environment: `conda activate authentication`

Trim Illumina adapters with Cutadapt:

```
for i in $(ls *.fastq.gz)
do
sample_name=$(basename $i .fastq.gz)
cutadapt -a AGATCGGAAGAG --minimum-length 30 -o ${sample_name}.trimmed.fastq.gz \ $
{sample_name}.fastq.gz -j 20
done
```

Taxonomic profiling with KrakenUniq

Download KrakenUniq complete microbial genomes RefSeq database:

```
wget https://figshare.scilifelab.se/ndownloader/articles/21299541/versions/1 \
&& export UNZIP_DISABLE_ZIPBOMB_DETECTION=true && unzip 1 && rm 1
```

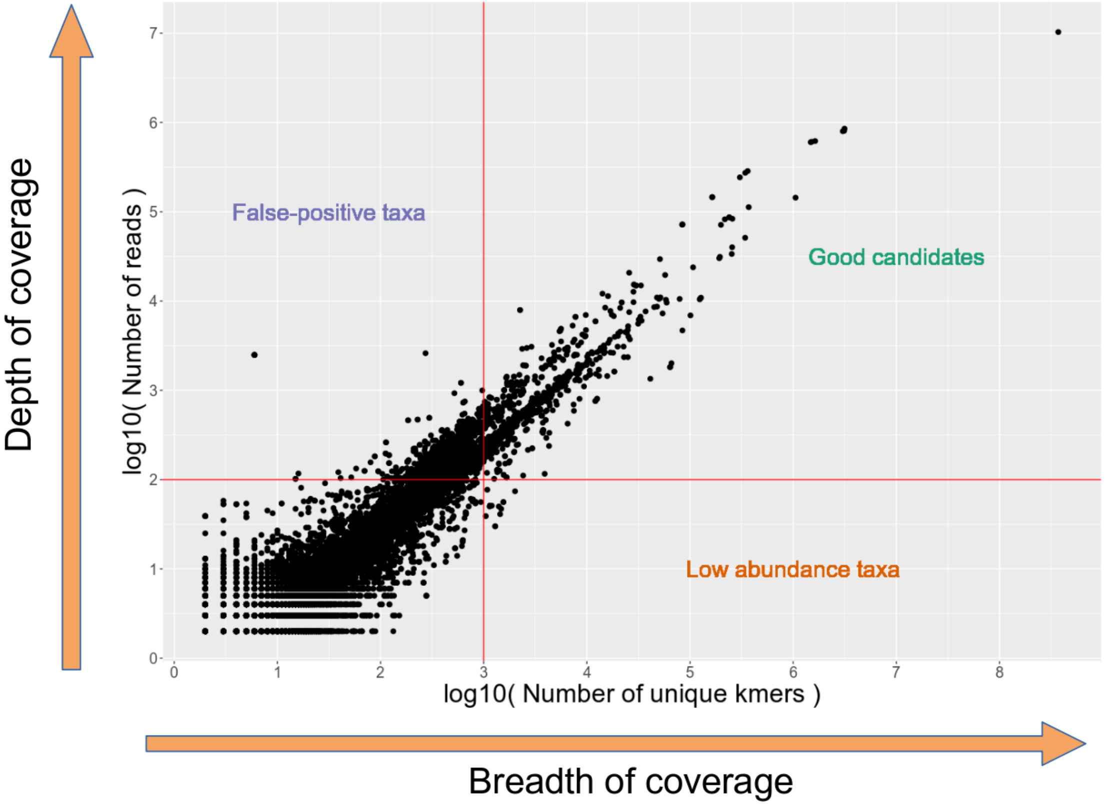
Run taxonomic kmer-based classification with KrakenUniq:

```
for i in $(ls *.trimmed.fastq.gz)
do
krakenuniq --db KRAKENUNIQ_DB --fastq-input $i --threads 20 \
--classified-out${i}.classified_sequences.krakenuniq \
--unclassified-out ${i}.unclassified_sequences.krakenuniq \
--output ${i}.sequences.krakenuniq --report-file ${i}.krakenuniq.output
done
```

KrakenUniq provides breadth of coverage info

We can filter KrakenUniq output with respect to both **depth** and **breadth** of coverage.

The **number of unique k-mers** per taxon provided by KrakenUniq is a proxy for breadth of coverage, and can be used for filtering out false positive findings.



Filter KrakenUniq output:*Y. pestis* in sample10

```
for i in $(ls *.krakenuniq.output)
do
$SCRIPTS_DIR/filter_krakenuniq.py $i 1000 200 $SCRIPTS_DIR/pathogenomesFound.tab
done
```

%	reads	taxReads	kmers	dup	cov	taxID	rank	taxName
1.523	11855	11553	164882	1.05	0.03772	632	species	Yersinia pestis
1.047	8151	7310	267081	1.06	0.03794	28450	species	Burkholderia pseudomallei
0.4386	3413	2560	49800	1	0.004508	28901	species	Salmonella enterica
0.4294	3342	749	36158	1.03	0.003238	305	species	Ralstonia solanacearum
0.2475	1926	1763	26882	1	0.01061	1314	species	Streptococcus pyogenes
0.08545	665	294	5727	1.05	0.0004944	587753	species	Pseudomonas chlororaphis

Build KrakenUniq abundance matrix

```
Rscript ${SCRIPTS_DIR}/krakenuniq_abundance_matrix.R KRAKENUNIQ \
KRAKENUNIQ_ABUNDANCE_MATRIX 1000 200
```

	sample1	**sample2**	**sample3**	**sample4**	**sample5**	**sample6**	**sample7**	**sample8**	**sample9**	**sample10**
Ralstonia solanacearum	3628	3751	619	1804	1384	1608	1375	0	1112	749
Mycobacterium avium	8236	8546	273	265	3221	4808	7750	6382	0	0
Burkholderia pseudomallei	7095	0	0	0	13082	0	4885	1456	0	7310
Salmonella enterica	4356	4471	4205	3588	0	13854	0	0	1959	2560
Pseudomonas chlororaphis	296	1024	0	977	374	677	276	0	0	294
Neisseria meningitidis	465	502	0	0	7341	0	0	3268	5643	0
Yersinia pestis	0	0	0	7174	957	0	11485	6461	0	11553

Follow up *Y. pestis* hit: compute alignments

Download *Yersinia pestis* reference genome from NCBI and build Bowtie2 index:

```
NCBI=https://ftp.ncbi.nlm.nih.gov; ID=GCF_000222975.1_ASM22297v1
wget $NCBI/genomes/all/GCF/000/222/975/${ID}/${ID}_genomic.fna.gz

gunzip ${ID}_genomic.fna.gz; echo NC_017168.1 > region.bed
seqtk subseq ${ID}_genomic.fna region.bed > NC_017168.1.fasta
bowtie2-build --large-index NC_017168.1.fasta NC_017168.1.fasta --threads 20
```

Align trimmed reads against *Yersinia pestis* reference genome with Bowtie2:

```
bowtie2 --large-index -x NC_017168.1.fasta --end-to-end --threads 20 \ --very-sensitive -U
sample10.trimmed.fastq.gz | samtools view -bS -h -q 1 \
-@ 20 - > Y.pestis_sample10.bam
samtools sort Y.pestis_sample10.bam -@ 20 > Y.pestis_sample10.sorted.bam
samtools index Y.pestis_sample10.sorted.bam
```

Compute evenness of coverage

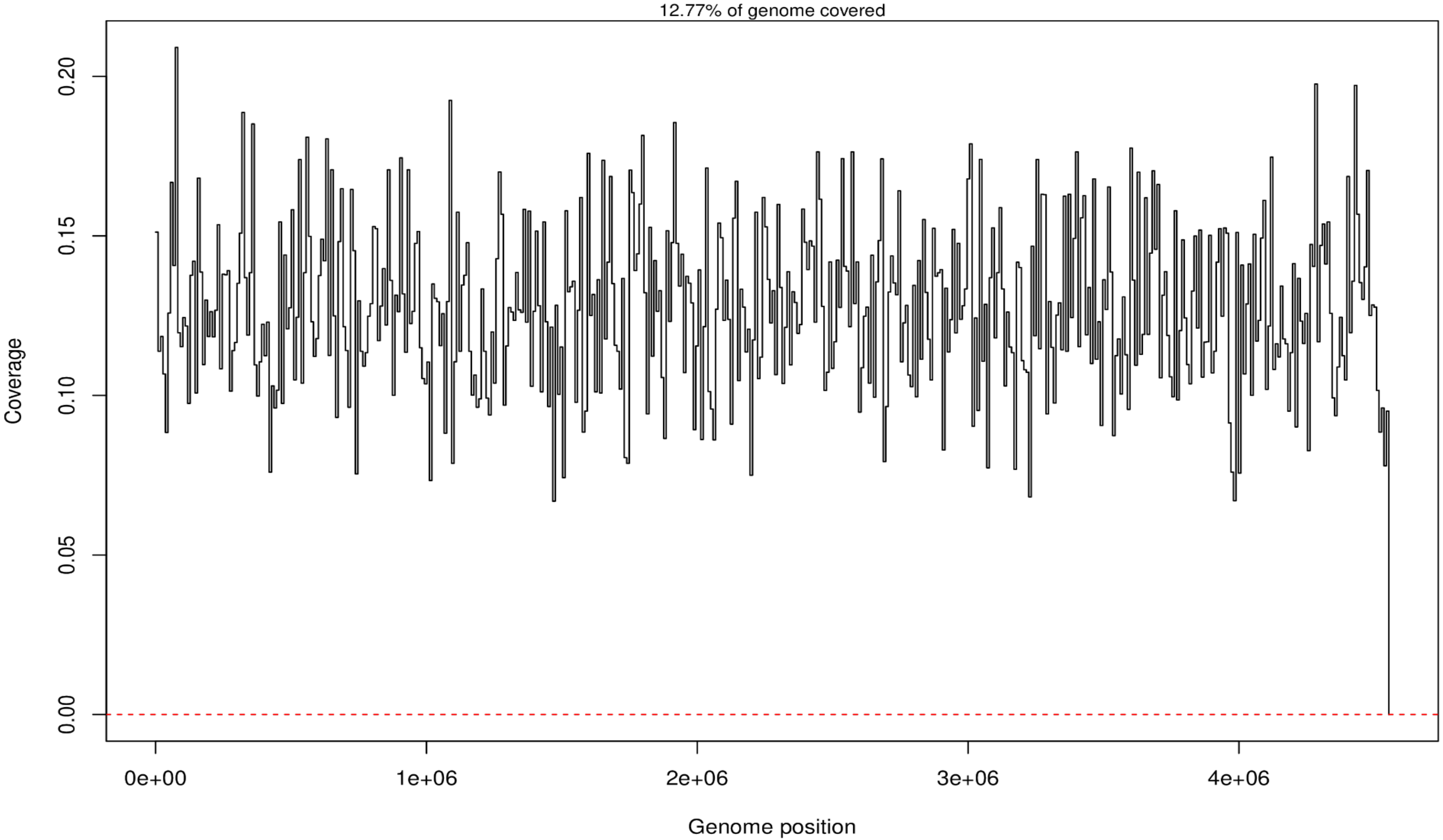
Compute breadth / evenness of coverage with *samtools depth*:

```
samtools depth -a Y.pestis_sample10.sorted.bam > Y.pestis_sample10.sorted.boc
```

Visualize evenness of coverage using the following R script:

```
df <- read.delim("Y.pestis_sample10.sorted.boc", header = FALSE, sep = "\t")
N_tiles <- 500; names(df) <- c("Ref", "Pos", "N_reads")
step <- (max(df$Pos) - min(df$Pos)) / N_tiles; tiles <- c(0:N_tiles) * step; boc <- vector()
for(i in 1:length(tiles))
{
  df_loc <- df[df$Pos >= tiles[i] & df$Pos < tiles[i+1], ]
  boc <- append(boc, rep(sum(df_loc$N_reads > 0) / length(df_loc$N_reads),
  dim(df_loc)[1]))
}
boc[is.na(boc)]<-0; df$boc <- boc
plot(df$boc ~ df$Pos, type = "s", xlab = "Genome position", ylab = "Coverage")
abline(h = 0, col = "red", lty = 2); mtext(paste0(round((sum(df$N_reads > 0) /
length(df$N_reads)) * 100, 2), "% of genome covered"), cex = 0.8)
```

Evenness of coverage: ~13% of genome covered

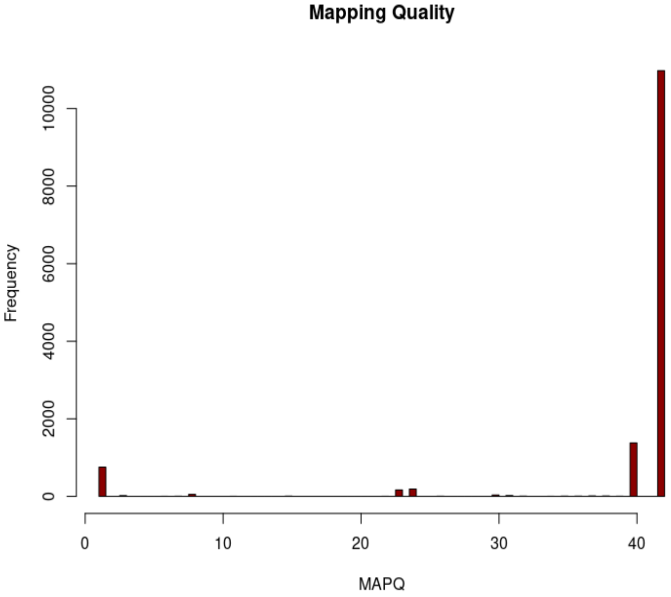


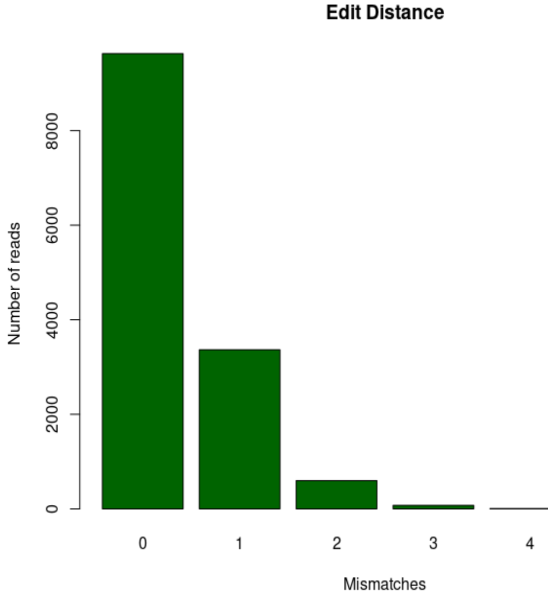
Assess alignment quality

Extract mapping quality and edit distance from BAM-alignments and visualize them:

```
library("Rsamtools"); par(mfrow = c(2, 1))
system("samtools view Y.pestis_sample10.sorted.bam | cut -f5 > mapq.txt")
hist(as.numeric(readLines("mapq.txt")), col = “darkred”, breaks = 100)
param <- ScanBamParam(tag = "NM"); bam <- scanBam("Y.pestis_sample10.sorted.bam", param = param)
barplot(table(bam[[1]]$tag$NM), col = ”darkgreen”, ylab="Number of reads", xlab = "Mismatches")
```

MAPQ score should be above 1 (multi-mappers have MAPQ = 0). **Edit distance** should have decreasing profile meaning high affinity of reads to reference

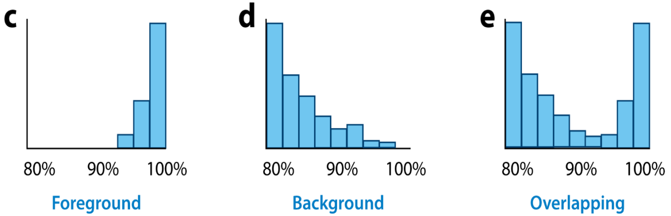


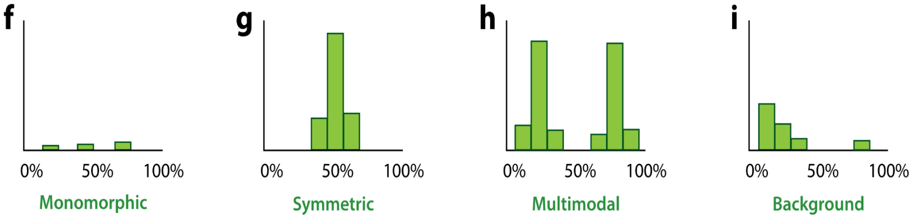


Affinity: percent identity and multi-allelic SNPs

Since bacteria are haploid organisms, only one allele is expected for each genomic position. Only a **small number of multiallelic sites** are expected, which can result from a few misassigned reads.

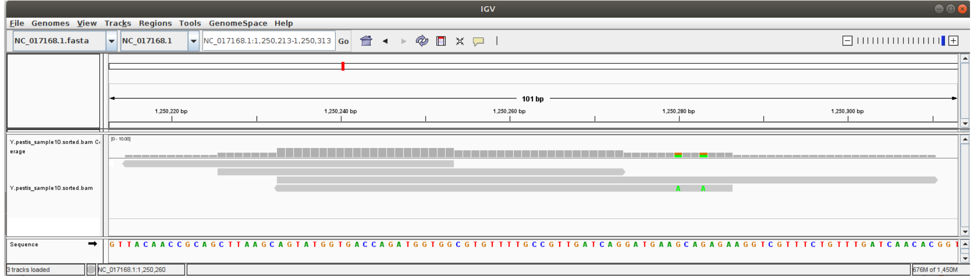
Percent identity



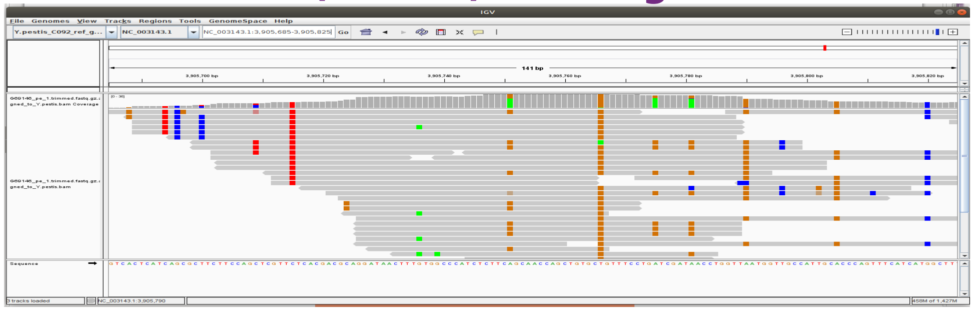


Warinner C, et al. 2017.
Annu. Rev. Genom. Hum. Genet. 18:321–56

Good alignments

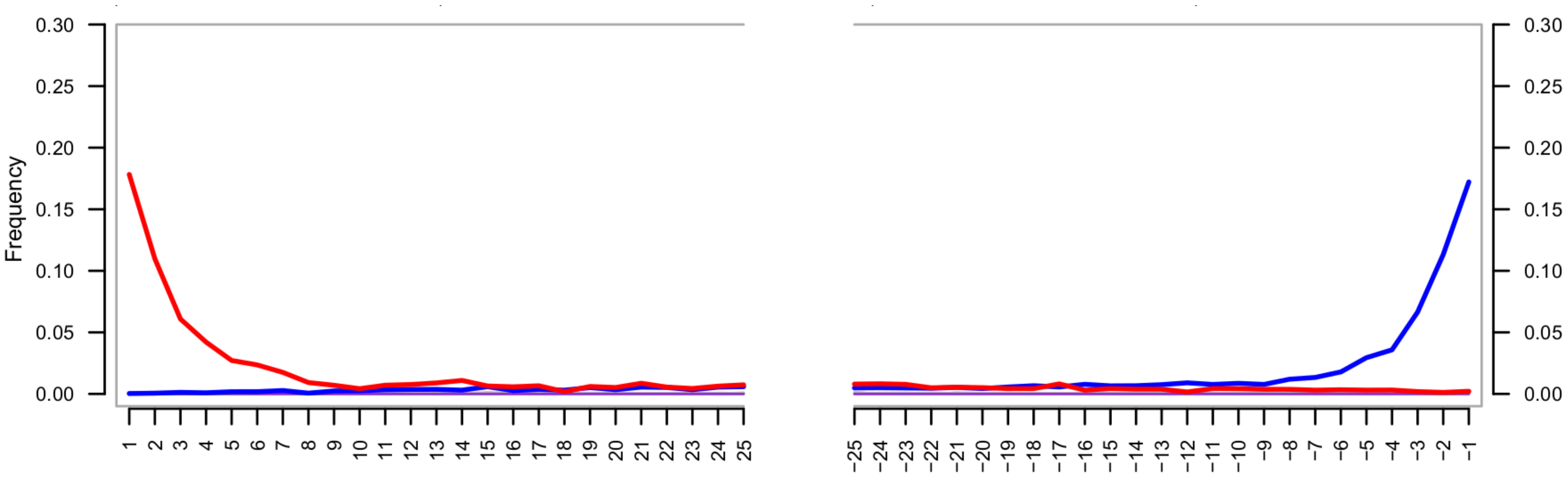

IGV
File Genomes View Tracks Regions Tools GenomeSpace Help
NC_017168.1.fasta
NC_017168.1
NC_017168.1:1,250,213-1,250,313
Go
101 bp
1,250,220 bp
1,250,240 bp
1,250,260 bp
1,250,280 bp
1,250,300 bp
Y.pestis_sample10.sorted.bam Coverage
[0 - 10.00]
Y.pestis_sample10.sorted.bam
A
A
Sequence
G T T A C A A C C G C A G C T T A A G C A G T A T G G T G A C C A G A T G G T G G C G T G T T T T G C C G T T G A T C A G G A T G A A G C A G A G A A G G T C G T T T C T G T T T G A T C A A C A C G G
3 tracks loaded
NC_017168.1:1,250,260
676M of 1,450M

(sort of) Bad alignments

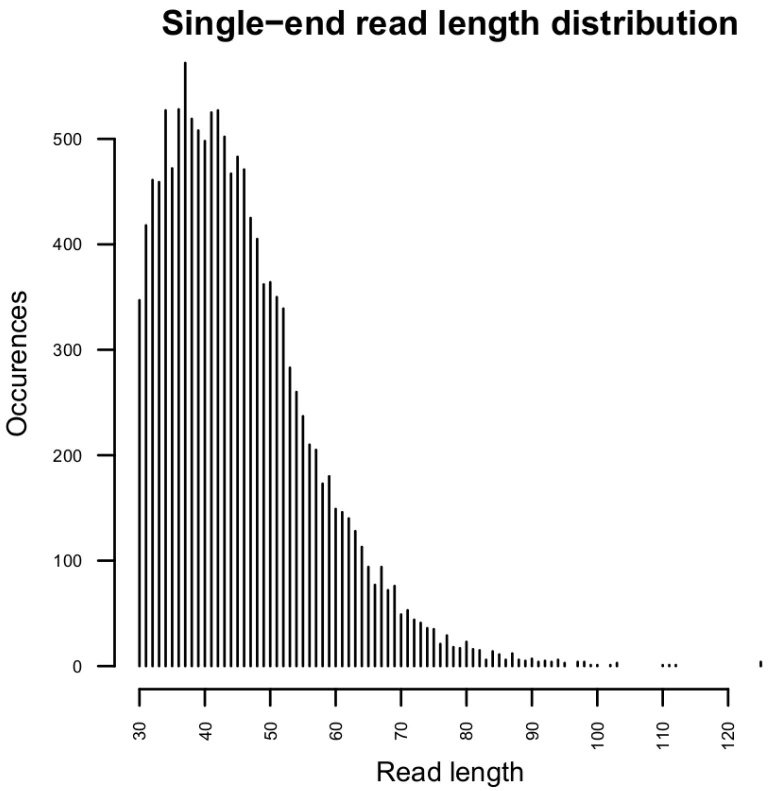

IGV
File Genomes View Tracks Regions Tools GenomeSpace Help
Y.pestis_C092_ref_g...
NC_003143.1
NC_003143.1:3,905,685-3,905,825
Go
141 bp
3,905,700 bp
3,905,720 bp
3,905,740 bp
3,905,760 bp
3,905,780 bp
3,905,800 bp
3,905,820 bp
G69146_pe_1.trimmed.fastq.gz.aligned_to_Y.pestis.bam Coverage
[0 - 36]
G69146_pe_1.trimmed.fastq.gz.aligned_to_Y.pestis.bam
Sequence
GTCACTCATCAGCGCTTCTTCCAGCTCGTTCTCACGACGCAGGATAACTTTGTGGCCCATCTCTTCAGCAACCAGCTGTGCTGTTTCCTGATCGATAACCTGGTTAATGGTTGCCATTGCACCCAGTTTCATCATGGCTT
3 tracks loaded
NC_003143.1:3,905,790
458M of 1,427M

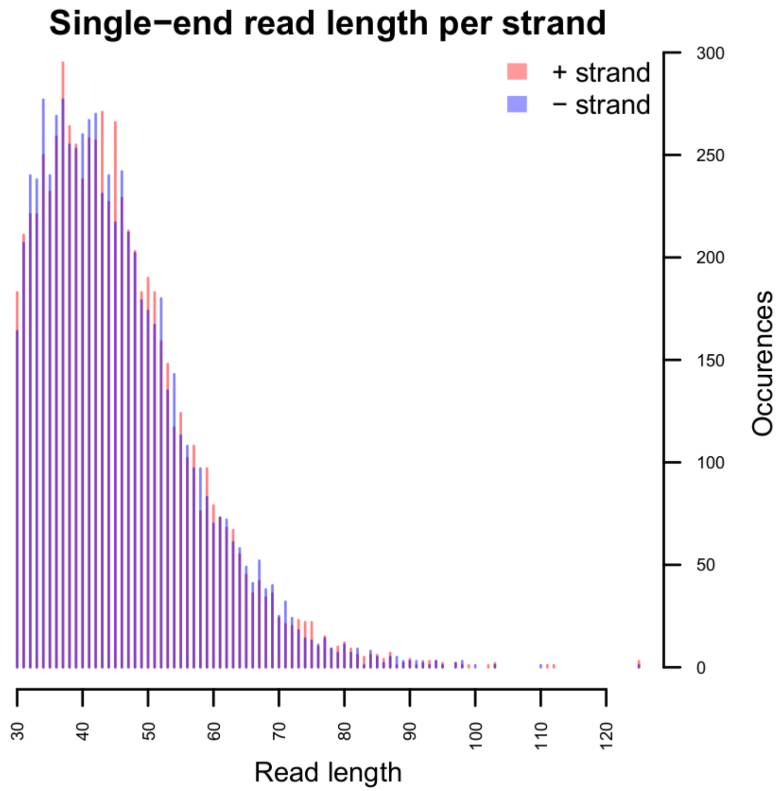
Compute deamination / damage profile

```
mapDamage -i Y.pestis_sample10.sorted.bam -r NC_017168.1.fasta -d MAPDAMAGE --merge-reference-sequences --no-stats
```



Read length distribution: DNA fragmentation





Compute distribution of PMD scores

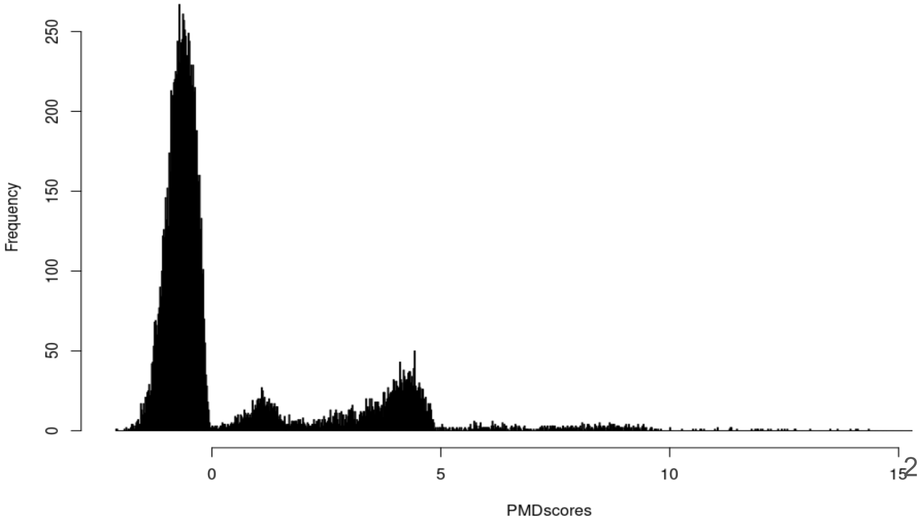
Compute PMD scores for each read with PMDtools:

```
samtools view -h Y.pestis_sample10.bam | python2 ${SCRIPTS_DIR}/pmdtools.0.60.py \ --printDS >
PMDscores.txt
```

Visualize PMD scores:

```
pmd_scores <- read.delim("PMDscores.txt", header = FALSE, sep = "\t")
hist(pmd_scores$V4, breaks = 1000, xlab = "PMD scores", main = "PMD Scores")
```

PMD scores are computed with **single read resolution** and are complementary to denomination profile. Substantial number of mapped reads should have **PMD scores above ~3**, this is a rule of thumb that can vary.



Deamination for damage-removed samples

The advantage of computing deamination profile with PMDtools is that it can compute deamination profile for **UDG / USER treated** samples. For this purpose, PMDtools uses only **CpG sites** which escape the treatment, so deamination is not gone completely and there is a chance to authenticate treated

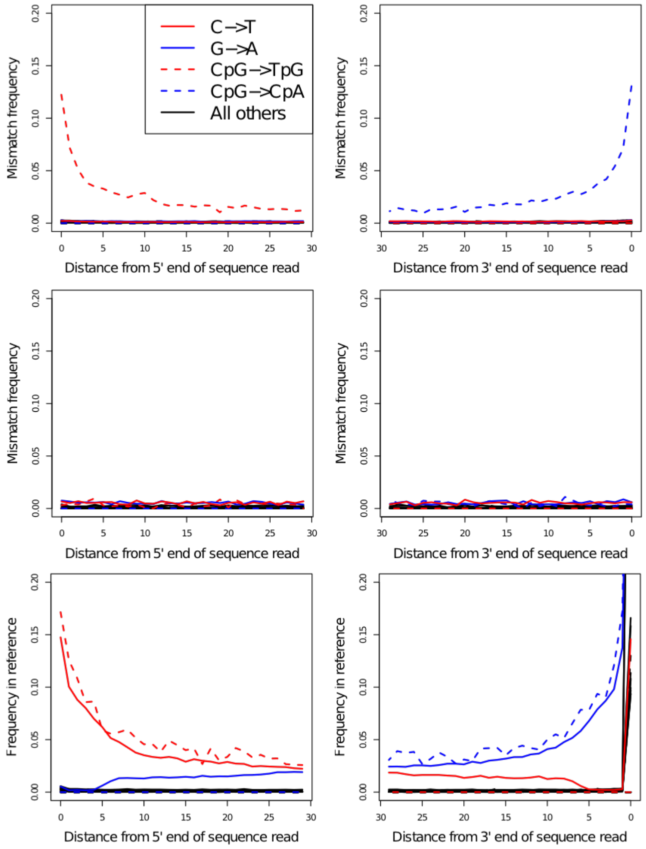
```
samtools view Y.pestis_sample10.bam | \ python2
${SCRIPTS_DIR}/pmdtools.0.60.py \ --platypus >
PMDtemp.txt

R CMD BATCH plotPMD.v2.R
```

Nuclear DNA USER-treated

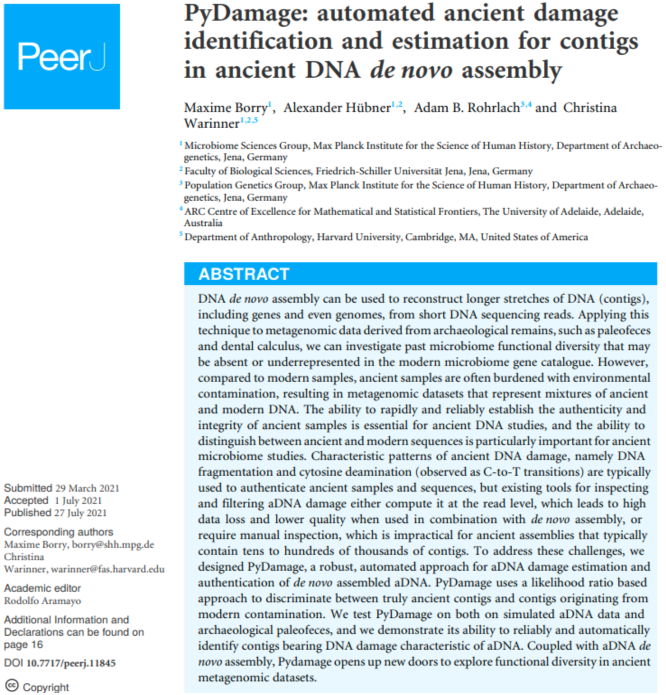
mtDNA USER-treated

No treatment



Authentication of *de-novo* assembled contigs

pyDamage evaluates the **amount of aDNA damage** and **tests the hypothesis** whether the model assuming presence of aDNA damage better explains the data than a null model



PeerJ

PyDamage: automated ancient damage identification and estimation for contigs in ancient DNA *de novo* assembly

Maxime Borry[1], Alexander Hübner[1,2], Adam B. Rohrlach[3,4] and Christina Warinner[1,2,5]

[1] Microbiome Sciences Group, Max Planck Institute for the Science of Human History, Department of Archaeogenetics, Jena, Germany
[2] Faculty of Biological Sciences, Friedrich-Schiller Universität Jena, Jena, Germany
[3] Population Genetics Group, Max Planck Institute for the Science of Human History, Department of Archaeogenetics, Jena, Germany
[4] ARC Centre of Excellence for Mathematical and Statistical Frontiers, The University of Adelaide, Adelaide, Australia
[5] Department of Anthropology, Harvard University, Cambridge, MA, United States of America

Submitted 29 March 2021
Accepted 1 July 2021
Published 27 July 2021

Corresponding authors
Maxime Borry, borry@shh.mpg.de
Christina
Warinner, warinner@fas.harvard.edu

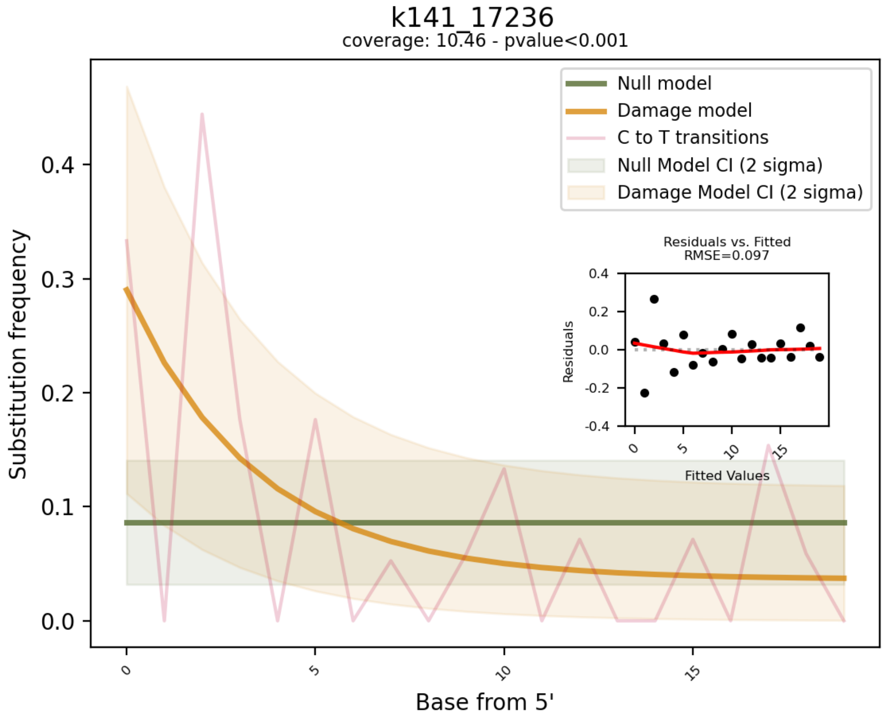
Academic editor
Rodolfo Aramayo

Additional Information and Declarations can be found on page 16

DOI 10.7717/peerj.11845

ABSTRACT

DNA *de novo* assembly can be used to reconstruct longer stretches of DNA (contigs), including genes and even genomes, from short DNA sequencing reads. Applying this technique to metagenomic data derived from archaeological remains, such as paleofeces and dental calculus, we can investigate past microbiome functional diversity that may be absent or underrepresented in the modern microbiome gene catalogue. However, compared to modern samples, ancient samples are often burdened with environmental contamination, resulting in metagenomic datasets that represent mixtures of ancient and modern DNA. The ability to rapidly and reliably establish the authenticity and integrity of ancient samples is essential for ancient DNA studies, and the ability to distinguish between ancient and modern sequences is particularly important for ancient microbiome studies. Characteristic patterns of ancient DNA damage, namely DNA fragmentation and cytosine deamination (observed as C-to-T transitions) are typically used to authenticate ancient samples and sequences, but existing tools for inspecting and filtering aDNA damage either compute it at the read level, which leads to high data loss and lower quality when used in combination with *de novo* assembly, or require manual inspection, which is impractical for ancient assemblies that typically contain tens to hundreds of thousands of contigs. To address these challenges, we designed PyDamage, a robust, automated approach for aDNA damage estimation and authentication of *de novo* assembled aDNA. PyDamage uses a likelihood ratio based approach to discriminate between truly ancient contigs and contigs originating from modern contamination. We test PyDamage on both on simulated aDNA data and archaeological paleofeces, and we demonstrate its ability to reliably and automatically identify contigs bearing DNA damage characteristic of aDNA. Coupled with aDNA *de novo* assembly, Pydamage opens up new doors to explore functional diversity in ancient metagenomic datasets.



Summary

- Evenness of coverage is an important metric for validation of findings

- Deamination profile, DNA fragmentation, mapping quality, edit distance and PMD scores are other authentication / validation metrics to consider