Norwegian University
of Life Sciences

**Master's Thesis 2019 30 ECTS**
Faculty of Chemistry, Biotechnology, and Food Science

# Recognizing plasmid-reads by machine learning and K-mer statistics

Jens Rasmus Liland
Chemistry and Biotechnology (M.Sc.) - Bioinformatics

# Abstract

The aim of this project was to investigate to what extent some machine learning methods are able to classify chromosome reads from plasmid reads based on K-mer statistics. Both short Illumina *HiSeq 2500* reads and medium-length Nanopore reads were simulated *in silico* from fully assembled *E.coli* chromosome and plasmid sequences. Both canonical and non-canonical K-mers were counted on all categories of sequence lengths. Working with *in silico* simulation data like this is different to a real-world experiment in that sequencing simulators like ART has arbitrary categorical simulation statistics, e.g. boolean presence of sequencing error, which were adjusted to find optimal combinations. K-mer methods worked great for fully assembled genome sequences, in terms of binary classification accuracy, decreasing substantually to 61 % for the Illumina sequences, while maintaining a fairly high level at 87 % for the Nanopore sequences. Wrongly classified reads mainly gets classified as plasmids. A 37X increase in sequence length[1] leads to a 42 % increase in accuracy[2].

[1] from 150 b to 5.8 kb
[2] from 61 % to 87 %

# Sammendrag

Målet med prosjektet var å undersøke i hvilken grad visse maskinlærings-metoder vil kunne klassifisere kromosom-reads fra plasmid-reads, basert på K-mer statistikk. Både korte Illumina *HiSeq 2500* reads, og mellomlange Nanopore reads ble simulert *in silico* fra komplett assemblerte *E.coli* kromosom- og plasmid-sekvenser. Både kanoniske og ikke-kanoniske K-merer ble talt for alle kategorier av sekvenslengder. Det å arbeide med *in silico* simulaterte data som disse er ulikt fra ikke-simulerte data ved at sekvens-simulatorer som ART har arbitrære, kategoriske simuleringsstatistikker, f.eks. boolsk tilstedeværelse av sekvenserings-feil, som ble justert for å finne optimale kombinasjoner. K-mer metoder fungerte veldig bra for fullstendig assemblerte genom-sekvenser, med hensyn til binær klassifikasjons-nøyaktighet, substansielt minkende til 61 % for Illumina-sekvensene, men opprettholder en temmelig nøyaktighet på 87 % for Nanopore-sekvensene. Feilklassifiserte reads blir hovedsakelig klassifisert som plasmider. En 37X økning i sekvenslengde[3] fører til en 42 % økning i nøyaktighet[4].

[3] fra 150 b til 5.8 kb
[4] fra 61 % til 87 %

# Table of Contents

# List of Figures

# List of Tables



# List of Code Listings



# Code availability. Programming appendix

The `R` and `python36` scripting pipeline built for the master thesis is available at multiple `git` repositories, both public and private (closed), on the Internet:

- `https://github.com/jenslila/liland2019master`, public.
- `https://github.uio.no/jensrli/liland2019master`, private.
- `https://bitbucket.org/jenslila/liland2019master/src/master/`, private.

# Glossary

**ABR** Antibiotic resistance. A subset of AMR, applying only to bacteria becoming resistant to antibiotics. 2

**ACC** Accuracy. Defined in equation 24 on page 17 as $\frac{TP+TN}{TP+TN+FP+FN}$. 17

**AMR** Antimicrobial resistance. 1

**ANOVA** ANalysis Of VAriance. 21, 22, 24, 26, 28, 31, 32

**ART** An open source set of *in silico* simulation tools for generating whole genome sequencing reads, `art_Illumina` being the binary for generating Illumina reads [1]. 1, 4–6, 28, 29

**BLAST** Basic Local Alignment Search Tool. 7, 14

**bp** base pairs. 12, 19

**CART** Classification And Regression Tree. 16

**cf.** *confer/conferatur*, latin abbreviations both meaning "compare", used to refer to other material when writing about a topic. 27, 34

**contig** Set of overlapping DNA segments that together represent a consensus region of DNA. 1, 2, 13, 35

**DeepSimulator** An open source *in silico* simulator of Oxford Nanopore reads [2]. 19, 28, 29

**e.g.** *exempli gratia*, latin abbreviation meaning "for example", indicating example scenario. i, 1, 4–6, 9, 11, 14–16, 18, 30, 34

**FA** Fully assembled whole genome sequences, i.e. fully assembled chromosomes and plasmids available from the NCBI Prokaryote database. v, vi, 2, 21, 28, 29, 34

**FN** False negative. 16, 29, 34

**FP** False positive. 16, 29, 32–34

**FreeBSD** A free and open source operating system descended from the Berkeley Software Distribution (BSD), which in-turn was based on Research Unix from Bell Labs. 5, 11

**git** A fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals. vi

**graphviz** A set of open-source tools initiated by AT&T Labs Research for drawing graphs specified in DOT language scripts. 3

**i.e.** *id est*, latin abbreviation for "that is", might be thought of as a forward pointing implication arrow. 2, 4–6, 9, 12, 15, 20, 28, 32

**IQR** The interquartile range. 18, 34

**KNN** K-nearest neighbour algorithm. 14, 31, 32

**Linux** A family of operating system distribution based on the free, open source, and Unix-like Linux kernel first released by Linus Thorvalds back in 1991. 5, 11

**Nanopore sequencing** A third generation approach used in the sequencing of biopolymers- specifically, polynucleotides in the form of DNA or RNA. i, ii, v, vi, 1, 19, 24–30, 33, 34

**NCBI** National Center for Biotechnology Information. 4, 5

**PCA** Principal component analysis, as provided by the R function `stats::prcomp`. 26, 34

**PCN** plasmid copy number. 6, 7, 12

$\Phi$ The Normal distribution's distribution function. Is solved in R using `stats::pnorm`. 18

$\Phi^{-1}$ The Normal distribution's quantile function. Is solved in R using `stats::qnorm`. 18

**PPV** Positive predictive value. Precision. Defined in equation 27 on page 17 as $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$. 17

**Python programming language** An interpreted, general-purpose programming language. Python can be said to be less abstracted (high-level) compared to the R programming language. 7, 8, 16

**R programming language** A programming language for statistical computing. 18

$\sigma$ standard deviation. Is solved in R using `stats::sd`. 18

**TN** True negative. 16, 29, 32, 34

**TNR** True negative rate. Specificity. Selectivity. Defined in equation 26 on page 17 as $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$. 17

**TP** True positive. 16, 29, 32, 33

**TPR** True positive rate. Sensitivity. Recall. Hit rate. Defined in equation 25 on page 17 as $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$. 17

**WGS** Whole Genome Sequencing. A sequencing process where the entire genome of an organism is sequenced without using methods for sequence selection. 1, 2, 5

# 1 Introduction

## 1.1 Antimicrobial resistance

Antimicrobial resistance (AMR) occurs in nature independent of human interaction [3]. Organisms harbouring antimicrobial resistance existed long before humans evolved, and have been found inside 30 000 year old Beringian permafrost, from long before humans evolved [3]. However, human interaction promotes the spread of antimicrobial resistant organisms, e.g. from poultry farms through squirrels back around to humans [4]. Antimicrobial resistance remains an increasing issue, e.g. annually causing more than 23 000 deaths out of two million infected people in the United States, mostly in healthcare settings [5]. Antimicrobial resistance mechanisms are often associated with plasmids in the bacterial cell making them easier to transfer between the cells [6].

## 1.2 Whole Genome Sequencing

Whole Genome Sequencing (WGS) is a a sequencing process where the entire genome of an organism is sequenced without using methods for sequence selection [7, page 338].

Redundancy of coverage was defined by Lander *et al.* [8] as the expected number of reads covering a given position on a genome. Given a sample, a sequencing machine is able to attain different levels of redundancy of coverage. If a plasmid occurs X times, this will become apparent after assembly in that contig islands belonging to the plasmid will have X times greater coverage compared to the chromosomal coverage. In these cases, plasmids and chromosomes can be easily distinguished based on coverage.

The traditional Illumina *HiSeq* 2500 sequencing machine uses a sequencing-by-synthesis cyclic reversible termination approach. Hundreds of millions of short templates copied from a sample are housed in clusters on a so-called flow cell. A mixture of 3′-blocked, individually labeled deoxynucleotides (dNTPs) are added to the flow cell. Fluorescent dyes and computer analysis of photographies of each of the clusters of the flow cell is used to identify the sequence of four dNTPs incorporated to each elongating complementary strand on the flow cell [7, 9].

Through `art_Illumina` [1] version 2.5.8, WGS reads of 11 Illumina WGS sequencing systems can be simulated *in silico*, the Illumina *HiSeq* 2500 being one such machine to choose among others. Specifying parameters such as redundancy of coverage, presence of a sequencing error profile, and read length to `art_Illumina` makes it possible to explore specific parameter combinations, e.g. exploring presence/absence of sequencing error profile.

Through the DeepSimulator [2], nanopore sequencing can be simulated *in silico*. The Nanopore technology is a new way of sequencing which we presume will become more and more common in the near future. The main differences from the Illumina data the ART `art_Illumina` simulates are

(1) Single reads instead of paired-end reads (no R1/R2 output file pairs)

(2) Much longer reads (thousands of bases)

(3) More sequencing errors, at least at present

## 1.3 K-mer frequencies

Methods of classification relies on numbers. Thus, when classifying text, such as nucleotide sequences, conversion from text to numbers will hopefully describe how the bodies of text differ from one another. One such method is to determine the occurence of K-mers in a sequence.

Given a nucleotide sequence of length L, it is possible to identify $L - K + 1$ number of nucleotide subsequences of length K. These nucleotide subsequences are also known as oligonucleotides, or K-mers. For a sequence composed of only the four nucleotide letters A, C, G, and T, there are $4^K$ number of unique K-mers. For odd values of K, the expected number of unique canonical K-mers becomes $4^K/2$, for even values of K, slightly higher due to palindromes. The sets of unique K-mers defines the dimensions (columns in a spreadsheet) of a dataset reporting the K-mer occurence of a set of nucleotide sequences (one data point [row] per sequence). Given a set of unique K-mers to look for, the number of K-mers in a nucleotide sequence can be identified. If the sequences in a set of sequences to be compared are of different lengths (different number of nucleotides within each nucleotide sequence), it makes sense to compare their K-mer frequencies, as opposed to comparing their K-mer counts, which would only indirectly be a comparison of sequence length, and not sequence structure in terms of K-mer occurrence. E.g. when comparing fully assembled (FA), whole chromosome and plasmid sequences, chromosome nucleotide sequences are always longer than plasmid nucleotide sequences, but when comparing reads from a WGS read simulation, the length of output sequences (reads) remains constant, removing the demand to calculate K-mer frequencies.

Although it is usual for K-mer based classification not to yield 100 % accurate classifications—at least when classifying using a neural network classifier from Google Brain's *TensorFlow* [10]—classifying WGS reads, as well as longer, partly assembled contigs, on the basis of K-mer frequencies has been shown to work better than redundancy of coverage, as well as sequence alignment [11].

## 1.4 Aim of this project

To be able to study the ABR mechanisms, we need to puzzle together the reads after sequencing, i.e. perform a sequence assembly. The reads we are faced with is a mixture of reads from the chromosome and the plasmids that may be there. The sequence assembly is challenging as long as we have such mixtures, especially when the plasmids have a low copy number, leading to a similar coverage as the chromosome. Thus, it would be very beneficial if we could separate reads that came from the chromosome from those from plasmids, based on K-mer statistics. The aim of this project is to investigate to what extent some machine learning methods are able to classify chromosome reads from plasmid reads based on such K-mer statistics.

# 2 Methods

The graphviz [12] flowchart in figure 1 maps out the processes of the scripting pipeline created for the project.

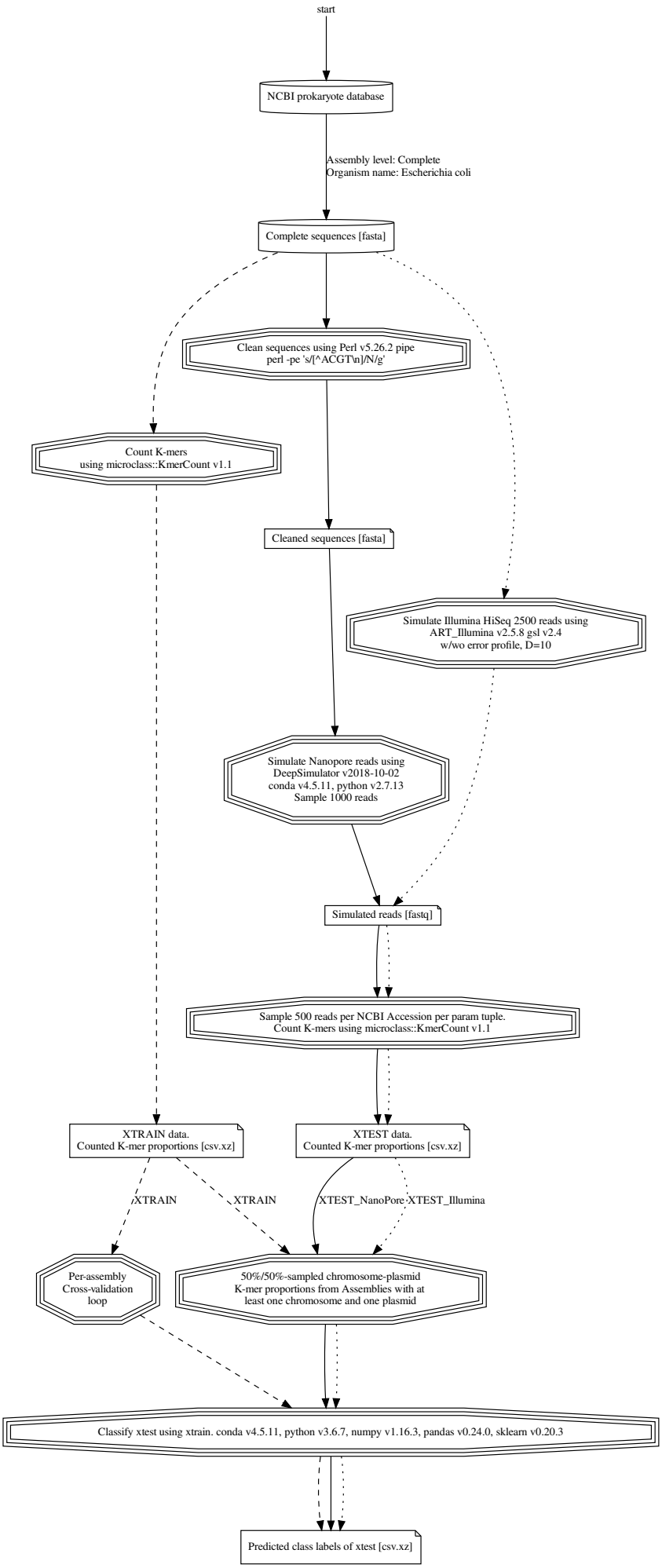



**Figure 1:** Flowchart of the simulation and classification pipeline

## 2.1 Data

We use simulated data in this project, because this allows us to create data sets where we know the origin of all reads, i.e. we know which reads are from a chromosome and which are from plasmids. In this way, we can evaluate methods by computing accuracies or other statistics revealing how close the classifications are to to the true statistics.

### 2.1.1 NCBI downloads and prepping

To train and test the model, data from 454 *Escherichia coli* genomes, were downloaded from the *National Center for Biotechnology Information*, NCBI, archives in the United States of America. An *E. coli* genome always contains one chromosome, and some genomes have one or more plasmids. We focused only on genomes with plasmids. A search of the NCBI prokaryote genome database website, `https://www.ncbi.nlm.nih.gov/genome/browse#!/prokaryotes/`, was done on 2019-04-07 using the criteria: (i) *organism name* set to *Escherichia coli*, and (ii) checkboxing *complete* in category *assembly level*. The search result was downloaded as a comma-separated values file (csv file). The results were filtered in R using using regular expressions. During filtering, it was assumed results with the word "plasmid" in the *Replicons* column has its chromosome listed first in the colon separated sequence of associated accession numbers. After determining all accession numbers, the associated sequences were downloaded using the R function `micropan::entrezDownload` [13].

Figure 2 shows how the plasmid count given the 454 chromosomes and 1219 plasmids downloaded from NCBI (blue crosshatch bars) at least exactly one chromosome, and less than 13 plasmids, i.e. a mean of $3 \pm 2$ plasmids per genome, with 0 %, 25 %, 50 %, 75 %, and 100 % number of plasmids quantiles of 1, 1, 2, 4, and 12, respectively. Some of the downloaded sequences contained characters which made it impossible to simulate *in silico* Whole Genome Sequencing reads from them, thus the whole genome was omitted. The red crosshatch dotted bars show the plasmid count in the new "ART" genome set, 1150 plasmids distributed over 433 genomes, e.g. the number of genomes with three plasmids in them decrease from 86 to 84. Quantile-wise, the decrease affects the 75 % quantile to decrease from 4 to 3.

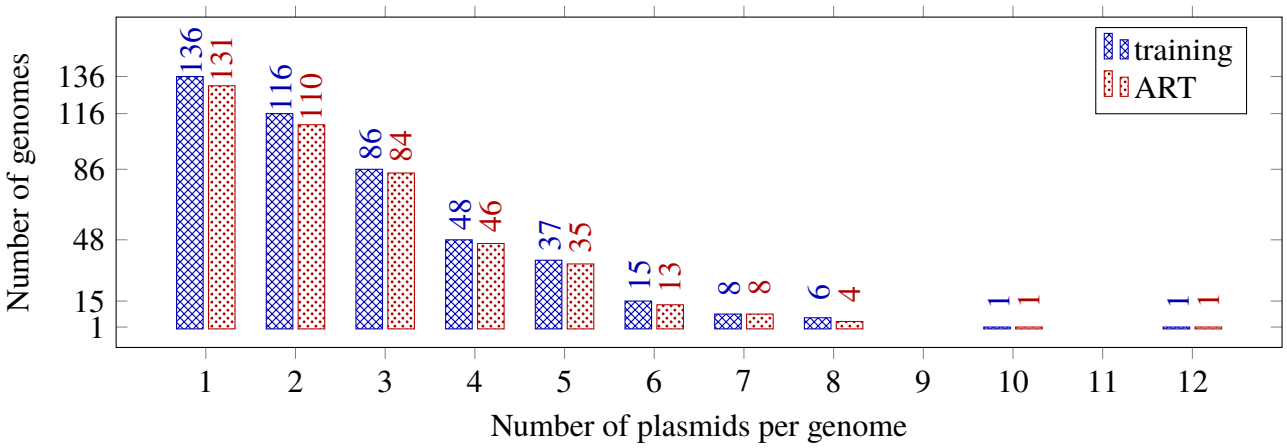



**Figure 2:** How the 1219 plasmids are distributed over the 454 genomes.

### 2.1.2 Discrimination based on whole chromosomes and plasmids

Classification based on relative K-mer frequency data of whole nucleotide sequences, as opposed to *in silico* WGS reads simulated from the completely assembled nucleotide sequences, is considered a "best-case scenario" in the sense that longer sequences of text is more probable to contain a greater number of subtle differences for precise classification. Classifying nucleotide sequences in a "best-case scenario" of long sequences makes sense to do before classifying shorter subsequences, to determine how great any upper threshold of classification accuracy can be attained, as the accuracy is expected to decrease when the length of nucleotide sequences decrease, as the difference in number of subtle differences decreases.

Each genome set was 454-fold cross-validated against the other 453 genomes, which means leaving out one genome "fold" to predict its response based on how its feature matrix matches to the feature matrix composed of all the 453 other genomes, in such a way that all the genomes are "left out" as a test set exactly once. As stated in section 2.1.1 on page 4, any genome chosen can not be expected to have a constant number of plasmids associated with it, greater than zero, less than 13; thus the number of rows of the feature matrix varies from 1660 to 1671 depending upon the number of nucleotide sequences (number of NCBI accession numbers) associated with the genomes of the "fold" being "left out."

Relative occurrence of both canonical K-mers and the complete set of K-mers were used for prediction.

### 2.1.3 Simulation of reads with ART

Given a nucleotide sequence and a set of parameters, the `art_Illumina` [1] simulation program can generate sequence read data of Illumina sequencers. As specified in the `art_Illumina` README updated on the 6th of June 2016, included within version 2.5.8 of the program suite, `art_Illumina` generates these reads "according to the empirical read quality profile summarized from large real read data."

Sequencing technology, or sequencing machine, i.e. sequencing system is specified using the `--seqSys` flag, e.g. `--seqSys HS25` indicates the Illumina *HiSeq 2500* machine.

The length of reads to be simulated by `art_Illumina` is declared by providing an integer to the `--len` flag, e.g. `--len 150` means a read length of 150 bp. When another operating system than Linux is used for running the simulation, e.g. FreeBSD/amd64 12.0-RELEASE, specifying `--len 150` might mean the resulting reads might end up being of a different length, e.g. 147 bp or 151 bp.

Paired-end reads can be simulated by `art_Illumina` providing the `--paired` boolean flag.

If fragment length in a paired-end simulation is assumed to be normally distributed, the mean and standard deviation of this normal distribution can be provided to `art_Illumina` through the `--mflen` and `--sdev` flags, respectively. The mean of the fragment length normal distribution is called the insert size.

The position of a fragment along the genome length ideally follows the uniform distribution, position $\sim$ uniform$(\alpha, \beta)$. The uniform distribution has two parameters, $\alpha$ and $\beta$, which can be thought

of as starting position and stopping position along a line. So along the genome length, parameter α might be though of as the first nucleotide in the genome, position 1, and parameter β then becomes the last nucleotide in the genome, position G. The probability density function of the uniform distribution is

$$f(x) = \frac{1}{\beta - \alpha},\ \alpha < x < \beta,\ -\infty < \alpha < \beta < \infty \tag{1}$$

Redundancy of coverage, also known as fold coverage or sequencing depth, can be specified in `art_Illumina` by providing an integer to the `--fcov` flag, e.g. `--fcov 100` means 100X coverage. Redundancy of coverage is explained further in section 2.3.1 starting on page 12.

As per equation 3 on page 12, the number of reads, N, is a combination of genome length, G, read length, L, and nucleotide coverage. Number of reads ideally attains either a binomial distribution, or a Poisson distribution. An alternative to specifying nucleotide coverage in `art_Illumina` is specifying the number of reads using the `--rcount` flag, which means nucleotide depth varies if genome length varies, as opposed to number of reads varying when genome length varies.

By providing the `--noALN` flag, `art_Illumina` does not output a ClustalW ALN alignment file.

`art_Illumina` uses the system time measured in seconds as the seed for its random number generator. As of version 2.5.8, the system time is obtained by including the first available `time.h` header file (at compilation)[5] and calling `time(NULL)` (at runtime), if no fixed seed was provided. A fixed seed can be provided to `art_Illumina` by using the `--rndSeed` flag.

`art_Illumina` ultimately outputs simulated reads in FASTQ format [16] to the file prefix provided to the `--out` flag. Each sequence is labeled with a unique FASTQ sequence identifier, e.g. if the FASTA title line in the input file in the path provided to the `--in` flag is

```
>CP009104.1 Escherichia coli strain RM9387, complete genome
```

the *prefix identification tag for read ID* provided to the `--id` flag is `data/ncbi/art/tmp`, a possible FASTQ sequence identifier could be

```
@CP009104.1-data/ncbi/art/tmp321840
```

To simulate reads without errors, the `--errfree` boolean flag is provided. The zero-error reads will end up in a SAM file, which has to be converted to R1 and R2 FASTQ files later using SAMtools [17] or similar tooling, to make the output in line with the regular (error containing) reads.

### 2.1.4 Plasmid copy number

An *E. coli* genome consists of a chromosome as well as a number of unique plasmids each having its own plasmid copy number (PCN), i.e. expected number of observed copies of a given plasmid within a cell. A plasmid having a low PCN is considered a rare plasmid, e.g. *Inc*I plasmids, which are potential carriers of antibiotic resistance, thus being important to discover. A low PCN implies low nucleotide

[5] which normally would be provided by the glibc library [14], if not a more exotic C library like musl [15] is used instead.

coverage, which makes low PCN plasmids hard to detect using nucleotide coverage statistics, which forms the basis of the plasmidSPAdes [18, page 3] algorithm (see page 12). A worst-case situation like this, where all of the possible plasmids of all of the known *E. coli* genomes have a PCN of 1, is the focus of this study.

### 2.1.5 Choice of Genomes

Due to the stacked prediction probability dataframe becoming dauntingly long for the 43 million combinations of

1000 classified reads per genome harbouring probabilities
× 454 genomes
× 3 K-mer levels
× 2 canonical levels
× 16 method levels

after classification of the simulated sequences, only classification results of the 50 first genomes[6] were further analyzed in the accuracy analysis as well as the prediction probability boxplot (which were further narrowed down to the intercept of each ANOVA accuracy analysis). That said, the classification of each genome were of course cross-validated towards all the other 453 genomes as they were classified. Keep this in mind when reading table 5 and 7 on pages 24 and 26.

## 2.2 K-mer statistics

Krawczyk et al. [11] showed in 2018 that classifying nucleotide sequences based on frequencies of a carefully selected set of oligonucleotides (K-mers) and deep neural network methods provided by Google Brain Team's software suite *TensorFlow* is able to provide a greater level of precision compared to previous BLAST (PlasmidFinder), coverage (Recycler), or sequence length (cBar) based methods, as long as the sequences to be classified are of sufficient length to be able to contain enough subtle differences to be distinguished from one another. In this project we decided to focus on the discrimination of plasmids from chromosomes based on K-mer statistics. In order to classify a DNA sequence, we need to convert the sequence information to numerical data in one way or another, since all classification methods are based on data as numbers. One way of converting text data to numerical data is to count K-mers. A string of length L can be sliced into

$$N = L - K + 1 \tag{2}$$

possible K-mers of length K. Three simple and straightforward ways of determined this set of K-mers might be either (1) through a Python programming language list comprehension in listing 1, (2) through a numpy vectorized anonymous function in listing 2, or (3) through a sapply call in R in listing

[6] thus reducing the number of combinations above to 4.8 million ...

3.

**Listing 1:** Finding the L-K+1 long set of K-mers in a string, seq, through Python programming language3 list comprehension.

```
[seq[i:(i+k)] for i in range(len(seq)-k+1)]
```

**Listing 2:** Finding the L-K+1 long set of K-mers in a string, seq, through a numpy vectorized anonymous function.

```
np.vectorize(lambda i: seq[i:(i+k)])(np.arange(len(seq)-k+1))
```

**Listing 3:** Finding the L-K+1 long set of K-mers in a string, seq, through a R sapply call.

```
sapply(seq(nchar(seq)-k+1), function(i) { substr(seq, i, i+k-1) })
```

### 2.2.1 Choice of K

Different values of K (word lengths) were tried out, as it is impossible to know the optimal word length. An upper limit of K = 8 was chosen, mostly due to memory usage and runtime. Word length, K, and four possible nucleotides yields $4^K$ possible combinations of K-mers, thus directly influencing memory usage and runtime.

### 2.2.2 Pseudo counts

To be able to calculate the binary log-ratio distance in equation 15 on page 15, the row from the training matrix, $\mathbf{X_0}$, can not contain zeroes, which would cause a division by zero, as well as the logarithm of zero remaining not applicable. To remedy this, a pseudo count, $\alpha$, can be added to every value of the count matrix right before calculating the relative K-mer occurrence. Adding a pseudo count of $\alpha = 0.5$ makes sense because 0.5 being half of a possible count-based increase of 1. Because whole plasmid sequences are shorter than chromosomes, a frequency previously being zero will after the pseudo count increase become greater than a zero from a chromosome.

### 2.2.3 Canonical K-mers

The main reason for counting canonical K-mers instead of counting all possible K-mers is to reduce the computing load the pipeline puts on the operating system, and hopefully use less runtime. Given a read and its reverse complentary sequence from a set of sequenced reads from a chromosome. Ideally, K-mers are to be counted in such a way that the K-mer frequency is equivalent for the read and its reverse complementary version, such that they will eventually be classified to the same object class. One way of doing this is to sum the K-mer count of the read and its reverse complement version. An alternative to this, would be to reduce any of the two possible sets of K-mers to the canonical set. E.g. given the sequence CATTC, the set of 2-mer counts becomes what is displayed in figure 3.

**Figure 3:** The set of 2-mer counts of the sequence CATTC.

| AT | CA | TC | TT |
|---|---|---|---|
| 1 | 1 | 1 | 1 |

Its reverse complementary sequence is GAATG, which has a set of 2-mer counts of what is displayed in figure 4.

**Figure 4:** The set of 2-mer counts of the sequence GAATG.

| AA | AT | GA | TG |
|---|---|---|---|
| 1 | 1 | 1 | 1 |

Thus the sum of the set of 2-mers for the sequence and its reverse complementary becomes what is displayed in figure 5.

**Figure 5:** The sum of the sets of 2-mer counts of the sequences CATTC and GAATG.

| AA | AT | CA | GA | TC | TG | TT |
|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 1 | 1 | 1 |

In comparison, the set of canonical 2-mers for both the sequence and its reverse complementary sequence becomes what is displayed in figure 6.

**Figure 6:** The canonical set of 2-mer counts for both of the sequences CATTC and GAATG.

| AA | AT | CA | GA |
|---|---|---|---|
| 1 | 1 | 1 | 1 |

As described in section 2.2.1, memory usage and runtime is directly influenced by the choice of K because it determines the number of K-mers. The canonical set of K-mers has fewer K-mer instances to be counted, thus reducing overhead when storing the counts in memory. The K-mers on the reverse complementary string is not necessary to count to achieve equivalence in K-mer counts between a sequence and its reverse complementary string, thus reducing runtime when counting.

For odd values of K, the number of unique canonical K-mers is exactly $4^K/2$. For even values of K, due to some K-mers being palindromes of their reverse complementary sequence, the number of unique canonical K-mers is greater than $4^K/2$, e.g. the reverse complementary sequence of the 4-mer CCGG is also CCGG. These two facts are illustrated in table 1 on page 10, and figure 7 on page 10.

A K-mer being its own palindrome is equivalent on both sides of the DNA backbone, no side being the canonical version in favour of the other. The count of a palindromic K-mer "not going anywhere", i.e. being unchanged from what it was in the complete set, can also be thought of as the palindromic count already being a sum of K-mer occurences, which thus can be expected to be greater than other K-mer occurences having a possible reverse complementary counterpart.

**Table 1:** How number of words, N, increases as word length, K, increases.

| K | $N_{canonical}$ | $N_{all}$ |
|---|---|---|
| 1 | 2 | 4 |
| 2 | 10 | 16 |
| 3 | 32 | 64 |
| 4 | 136 | 256 |
| 5 | 512 | 1024 |
| 6 | 2080 | 4096 |
| 7 | 8192 | 16384 |
| 8 | 32896 | 65536 |

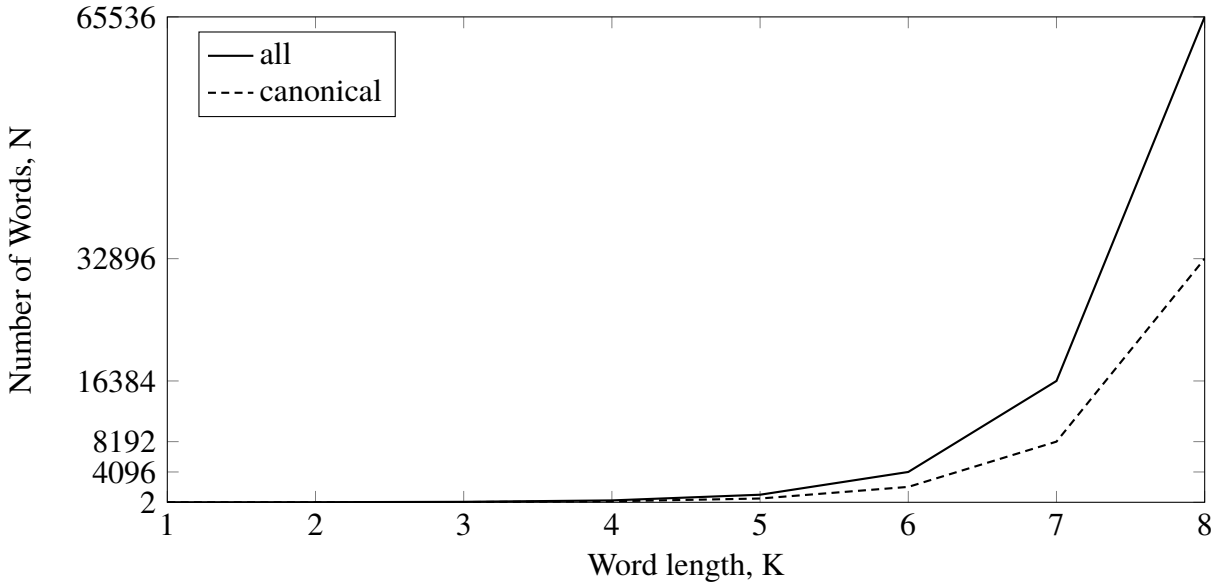



**Figure 7:** How number of all possible words and canonical words increases as word length increases.

### 2.2.4 Efficient multi FASTA K-mer counting using `fastatuples.c`

`fastatuples.c`[7] is a portable, 1403 lines of code long, C program which works on both the authors Linux version 5.0.2 laptop and his FreeBSD/amd64 12.0-RELEASE r341666 "counting" server. On Linux at least, the C program's only dependencies seems to be glibc [14], e.g. `string.h`. `fastatuples.c` is simple to compile anywhere using an arbitrary version of the GNU C compiler by issuing something like

**Figure 8:** Compiling `fastatuples.c` using an arbitrary version of the GNU C compiler.

```
gcc -Wall -std=c99 -pedantic -O3 -o fastatuples fastatuples.c
```

`fastatuples.c` is a very flexible program in that it reads multi FASTA input from standard input, meaning it can be used in a UNIX pipeline like

**Figure 9:** Multi FASTQ 8-mer counting using `fastatuples.c` in a UNIX pipeline.

```
xz -cd CP012633.1_errFree.R1.fq.xz | \
  sed '/^@/!d;s//>/;N' | \
  fastatuples -w 8 -s | \
  sort | \
  uniq -c | less
```

The `-w` flag is mandatory, and specifies K-mer length, e.g. `-w 8` means 8-mers will be extracted. The `-s` flag is optional, and includes the sequence index before every word, e.g. a line `17AGGATAAA` means the 8-mer AGGATAAA was found in sequence number 17 in the input FASTA lines.

[7] `fastatuples.c` was written in 2016 by David Mathog <mathog@caltech.edu> of the Biology Division at The California Institute of Technology, and is made publicly available at http://saf.bio.caltech.edu/pub/software/molbio/fastatuples.c. Version 1.0.19 was released on 2017-02-03.

## 2.3 Classification

### 2.3.1 Redundancy of coverage

Redundancy of coverage is the expected number of reads covering a given position on a genome, and was defined by Lander and Waterman back in 1988 [8, page 232] [19] as

$$c = LN/G \tag{3}$$

where c is nucleotide coverage, L is read length in base pairs (bp), N is number of reads, and G is genome length in bp. E.g. when sequencing N number of reads of length $L = 300\,\text{bp}$ of an *E. coli* genome, G, consisting of just one chromosome (no additional plasmids) of length $C = 5\,\text{Mbp}$, the coverage becomes $6 \times 10^{-5}N$ because

$$c = \frac{300\,\text{bp} \cdot N}{5\,\text{Mbp}} = 6 \times 10^{-5}N \tag{4}$$

Likewise, for a plasmid of length $P = 0.1\,\text{Mbp}$, the coverage becomes $3 \times 10^{-3}N$ because

$$c = \frac{300\,\text{bp} \cdot N}{0.1\,\text{Mbp}} = 3 \times 10^{-3}N \tag{5}$$

which means $\partial G < 0 \iff \partial N < 0$, i.e. as genome length G decreases, a lower number of reads N is needed to obtain the same coverage.

If a plasmid has a PCN greater than 1, this becomes reflected in the number of reads if nucleotide coverage, genome length, remains constant, i.e.

$$\partial PCN > 0 \implies \partial N > 0,\ \partial G = \partial c = 0,\ PCN > 1 \tag{6}$$

Continuing the *E. coli* example from above, when the genome additionally consists of 10 copies ($PCN = 10$) of a plasmid of length $P_1 = 0.1\,\text{Mbp}$, the PCN does not reflect the total length of the genome, the length of the genome increases 2 % only by the 0.1 Mbp plasmid length to 5.1 Mbp because

$$G = \chi + P_1 = 5\,\text{Mbp} + 0.1\,\text{Mbp} = 5.1\,\text{Mbp} \tag{7}$$

thus the coverage decreases 1.7 % by $6 \times 10^6 N$ to $5.9 \times 10^{-5}N$ because $c = \frac{300\,\text{bp} \cdot N}{5.1\,\text{Mbp}} = 5.9 \times 10^{-5}N$.

### 2.3.2 Decision rule

A decision rule, $\delta(\theta)$, is defined [20, page 376] as a function of an observed value of a statistic becoming a point estimate of a parameter, $\theta$, of a statistical distribution, $f(x;\theta)$, of the discrete or continuous type.

### 2.3.3 The decision rule of plasmidSPAdes

The decision rule, $\delta_c(\mathcal{E})$, in plasmidSPAdes [18, page 3] for a long edge, $\mathcal{E}$, in the assembly graph being classified as a chromosomal edge, $\chi$, is

$$\delta_\chi(\mathcal{E}) = \frac{\text{Coverage}(\mathcal{E})}{\text{medianCoverage}} \in (1 \pm \mathcal{D}) \tag{8}$$

where $\mathcal{D}$ is the maximum deviation, defaulting to 0.3, and medianCoverage is the maximum coverage for which the collection of all long contigs of that coverage or greater covers at least half of the total length of the collection of all long contigs in the SPAdes assembly graph. [18, page 2]

### 2.3.4 Response and predictors

The term *response* is used in the modern language of machine learning and pattern recognition literature to describe what in classic statistical literature is known as a *dependent variable* [21, page 9]. Likewise, a *predictor* or *feature* indicates what was classically known as an *independent variable*.

### 2.3.5 Prediction probability

A prediction probability signifies how sure the classifier is on the predicted class (response). E.g. for a random forest classifier using 100 trees, a prediction probability of 0.68 means 68 trees voted for the predicted class. A practical example would be to think of a case where one asks random passersby in a foreign, unfamilliar town for the way to specific locations (the predicted class), and then asks a second question of how sure the person is that his answer is correct (the prediction probability).

### 2.3.6 Training set and test set

A *training set* is a set of example responses and predictors used to fit a classifier to predict the responses of new unseen objects with the same predictors. [21, page 1 and 2] A *test set* should ideally be some known example observations[8] independent of the training and validation sets, used at the end of the data analysis, after a model has been selected, to assess the generalization error. [21, page 222]

### 2.3.7 Evaluation of classifiers, cross-validation

Cross-validation is a method for estimating prediction error. By splitting scarce[9] training data into K number of roughly equally sized parts, calculating the prediction error for every $k$th part based on a model fitted to the other K − 1 parts of the training data, a combined prediction prediction error can be established [21, page 241 to 243].

[8] E.g. 25 % of the original example data, the rest 75 % used for training and test set during model selection
[9] having few observations

## 2.4 Selected methods

### 2.4.1 K-Nearest Neighbours and distances

A Bayes classifier [22, page 52] assigns a test observation with predictor vector $\mathbf{x_0}$ to a class $j$ which maximizes the conditional probability

$$\mathrm{P}(\mathrm{Y} = j|\mathrm{X} = \mathbf{x_0}) \tag{9}$$

e.g. for two classes, `chromosomes` or `plasmids`, $\mathrm{Y} \in \{\texttt{chromosomes}, \texttt{plasmids}\}$, class `chromosomes` is predicted if $\mathrm{P}(\mathrm{Y} = \texttt{chromosomes}|\mathrm{X} = \mathbf{x_0}) > 0.5$, and class `plasmids` otherwise. The conditional probability equation 9 can be estimated through a number of methods, one of easiest to understand being the K-nearest neighbour (KNN) classifier. A lower threshold of the Bayes classifier can be set to reduce the number of false predictions by not classifying any prediction below the threshold, e.g. Krawczyk et al. [11, page 7] does this to compare their proposed pipeline PlasFlow (kmer frequency based) to cBar [23] (nucleotide sequence length based), PlasmidFinder [24] (BLAST based), Recycler [25] (coverage based), and others.

This paragraph explains how equation 9 can be predicted using a non-parametric method such as the KNN classifier. As per James et al. [22, page 39], the KNN Bayes classifier prediction is a two-step process, (1) given the neighbour count, $\mathrm{K} > 0$, and training data, $\mathbf{X}$, a "neighbourhood," $\mathcal{N}_0$, is identified as the K number of points closest to $\mathbf{x_0}$; (2) the conditional probability for class `chromosomes` becomes the fraction of points in the neighbourhood, $\mathcal{N}_0$, whose response equals `chromosomes`, otherwise class `plasmids` is predicted. Written in condensed mathematical notation this becomes:

$$\mathrm{P}(\mathrm{Y} = \texttt{chromosomes}|\mathrm{X} = \mathbf{x_0}) = \mathrm{K}^{-1} \sum_{i \in \mathcal{N}_0} \mathrm{I}(\mathrm{y}_i = \texttt{chromosomes}) \tag{10}$$

otherwise class `plasmids` is predicted, where $\mathrm{y}_i$ is the response of a neighbour in the neighbourhood, $\mathcal{N}_0$, $\mathrm{I}(\mathrm{y}_i = \texttt{chromosomes})$ is an indicator variable [22, page 37] equaling 1 if $\mathrm{y}_i = \texttt{chromosomes}$, and zero if $\mathrm{y}_i \neq \texttt{chromosomes}$, much like when converting a boolean value to a numeric value in R[10] will return either 0 or 1. What we see in equation 10 is the response of an unkown point of the test set being predicted based on a majority vote of what the K neighbouring points identify themselves as; the proportion of votes for each of the classes becoming their conditional probability. The distances from the unknown point of the test set to all of the points in the training set to determine the K nearest points belonging to the neighbourhood $\mathcal{N}_0$ can be calculated according to a number of different metrics, the most popular ones being Euclidean, Manhattan, and Chebyshev.

The Minkowski distance between two points X and Y becomes

$$\mathrm{D}_{\mathrm{Minkowski}}(\mathrm{X},\ \mathrm{Y}) = \left( \sum_{i=1}^{\mathrm{n}} \left| \mathrm{x}_i - \mathrm{y}_i \right|^{\mathrm{p}} \right)^{1/\mathrm{p}} \tag{11}$$

[10] E.g. `as.numeric("chromosomes"=="plasmids")` returns `0` (numeric zero)

where p is the power parameter, a constant integer. Given values of p, the Minkowski distance defines a family of distance metrics consisting of Manhattan, Euclidean, and Chebyshev; i.e. Manhattan distance for p = 1, Euclidean for p = 2, and Chebyshev distance when $p \to \infty$. Thus the Manhattan distance between two points X and Y becomes

$$D_{\text{Manhattan}}(X,\ Y) = \sum_{i=1}^{n} \left| x_i - y_i \right| \tag{12}$$

The Euclidean distance between two points X and Y becomes

$$D_{\text{Euclidean}}(X,\ Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{13}$$

The Chebyshev distance between two points X and Y becomes

$$D_{\text{Chebyshev}}(X,\ Y) = \lim_{p \to \infty} \left( \sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} = \max_{i=1}^{n} (|x_i - y_i|)^2 \tag{14}$$

A fifth distance metric, unrelated to the Minkowski distance, a binary log-ratio distance becomes

$$D_{\text{LogRatio}}(X,\ Y) = \sum_{i=1}^{n} \left| \log_2 \frac{y_i}{x_i} \right|,\ x_i \neq 0 \tag{15}$$

$x_i$ can never be 0, as division by zero is mathematically undefined, causing the computation to *segmentation fault*. This can be resolved using pseudo counts. If the use of the binary log-ratio distance metric enables a successful classification using the K-nearest neighbour classifier, this indicates that rare K-mers are more important than frequent occurring K-mers for separating classes, e.g. given two points of two dimensions, $[0.05, 0.50]$ and $[0.10, 0.40]$, the distance from 0.05 to 0.10 in the first dimension becomes greater than from 0.50 to 0.40 in the second dimension with the binary log ratio, but lesser with the Euclidean distance, because

$$\left| \log_2 \frac{0.05}{0.10} \right| > \sqrt{(0.05 - 0.10)^2} \tag{16}$$

$$\left| \log_2 \frac{0.50}{0.40} \right| < \sqrt{(0.50 - 0.40)^2} \tag{17}$$

### 2.4.2 Random forests

A random forest classifier can be defined as [26, page 6] a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k),\ k = 1, \ldots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors, and each tree casts a unit vote for the most popular class at input $\mathbf{x}$.

From training data, a random sample with replacement is drawn with replacement (an element may appear more than once in the sample), forming the building blocks of every tree.

The randomness in the random forest classifier is due to a subset of m number of variables being

chosen from p number of possible features for to determine the best variable to split from [21, page 588]. In classification problems with p number of features, the m number of features chosen at each split is typically $\sqrt{p}$, but can be as low as 1 [21, page 589].

In the Classification And Regression Tree (CART) [27] decision tree algorithm, within each tree of the random forest, the variable with the lowest node impurity, provides the best split of a node into two daughter nodes, given the set of m possible varibles to split from. For classification problems, node impurity can either be measured through the Gini impurity, or cross-entropy/information gain/Shannon entropy [21, page 309]. At version 0.20.3 of the scikit-learn Python programming language library, its random forest classifier (`sklearn.ensemble.RandomForestClassifier`) defaults to using the Gini impurity to determine the quality of split.

The impurity function, $i_G(\mathbf{x})$, based on the Gini index [28] [21, page 309] [29, page 45] is

$$i_G(\mathbf{x}) = \sum_{k=1}^{J} P(c_k|\mathbf{x})[1 - P(c_k|\mathbf{x})] \tag{18}$$

The impurity function, $i_H(\mathbf{x})$, based on the cross-entropy/information gain/Shannon entropy [30] [21, page 309] [29, page 45] is

$$i_H(\mathbf{x}) = -\sum_{k=1}^{J} P(c_k|\mathbf{x}) \log_2[P(c_k|\mathbf{x})],\ P(c_k|\mathbf{x}) \neq 0 \tag{19}$$

In both equation 18 and 19, $P(c_k|\mathbf{x})$ is the probability of an item with label $c_k$ being chosen among J number of possible labels (J = 2 in this case, $c_k \in \{\texttt{chromosomes}, \texttt{plasmids}\}$).

## 2.5 Analysis

### 2.5.1 The 2x2 confusion matrix—the end product of the binary classification test

For the binary classification of this text, let the plasmid class be the positive class (P) among the two classes plasmids and chromosomes, chromosomes thus being the negative class (N). A vector of correct responses for a set data points of to be classified is available. Correct response in hand, the *true positives* (TP) is the number of plasmids correctly classified as a plasmid. In a traditional binary classification sense, where the positive class is actively sought after, e.g. being a diagnosis of some sort, the TP is also known as the number of hits (being achieved by a given test). The *false positives* (FP) is the number of plasmids that was falsely classified as a chromosomes. Also known as the number of false alarms, or numbers of Type I errors. The *true negatives* (TN) is the number of chromosomes correctly classified as a chromosome. Also known as the number of correct rejections. The *false positives* (FN) is the number of chromosomes falsely classified as a plasmid. Also known as the number of misses, or number of Type II errors. These four numbers[11] are usually displayed in a $2 \times 2$ confusion matrix, like the one in table 2.

[11] TP, FP, TN, and FN

**Table 2:** $2 \times 2$ confusion matrix for classification of chromosomes and plasmids

| predicted \ true | plasmids | chromsomes |
|---|---|---|
| plasmids | TP | FP |
| chromosomes | FN | TN |

### 2.5.2 Accuracy, sensitivity, specificity, and precision—metrics for comparison of 2x2 confusion matrices

To analyze a $2 \times 2$ confusion matrix, the result of binary classification problem, accuracy, sensitivity, and specificity are common measures. Precision being a fourth.

Accuracy (ACC), equation 24, is the proportion of correctly classified data points among all possible data points. In this case: the proportion of correctly classified plasmids and chromosomes among all sequences in the test set. In a medical setting: the proportion of sick and healthy patients correctly diagnosed with a specific condition.

Sensitivity, equation 25, also known as recall, hit rate, as well as the true positive rate (TPR), is the proportion of true positives among all positives. In this case: the proportion of correctly classified plasmids among all possible plasmids. In a medical setting: the proportion of sick patients correctly classified as having a specific condition.

Specificity, equation 26, also known as selectivity, and the true negative rate (TNR), is the proportion of true negatives among all negatives. In this case: the proportion of correctly classified chromosomes among all possible chromosomes to be classified. In a medical setting: the proportion of healthy people correctly classified as not having a specific condition.

Precision, equation 27, also known as the positive predictive value (PPV), is the proportion of true positives among positives. In this case: the proportion of actual plasmids among data points classified as plasmids. In a medical setting: the proportion of patients actually having a condition among sick and healthy patients diagnosed with the condition. Combinations of low/high bias/variance in the case of a shooting disc is illustrated in figure 10.

$$\mathrm{TP} = \text{true positives} = \text{hit} \tag{20}$$

$$\mathrm{TN} = \text{true negatives} = \text{correct rejection} \tag{21}$$

$$\mathrm{FP} = \text{false positives} = \text{false alarm} = \text{Type I error} \tag{22}$$

$$\mathrm{FN} = \text{false negatives} = \text{miss} = \text{Type II error} \tag{23}$$

$$\mathrm{Acc} = \text{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{24}$$

$$\mathrm{TPR} = \text{sensitivity} = \text{recall} = \text{hit rate} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{25}$$

$$\mathrm{TNR} = \text{specificity} = \text{selectivity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{26}$$

$$\mathrm{PPV} = \text{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{27}$$

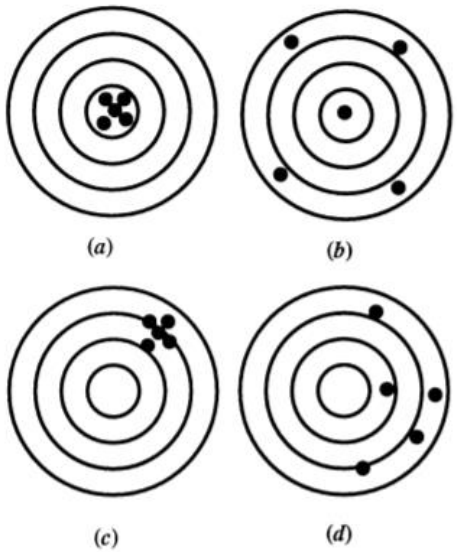



**Figure 10:** Shooting disc scenarios showing the four combinations of low/high variance/bias [31]. (*a*) shows low variance and low bias, (*b*) shows high variance and low bias, (*c*) shows low variance and high bias, and (*d*) shows high variance and high bias. It can be said that (*c*) is a case of underfitting, while (*b*) as well as (*d*) might be called cases of overfitting.

### 2.5.3 Boxplots

A boxplot is a way to visualise, and thus more easily compare to another sample, the median, quartiles, and outliers of a sample of data points. The interquartile range (IQR), is the difference between the 75th and 25th percentiles, representing 50 % of the data points, illustrated in the plot by a box. Inside the box, the sample's median value is marked by a broad line. Outside the box, a set of whiskers might be observed. The whiskers might represent different things, e.g. the 0th and 100th percentile. In the R function `graphics::boxplot`, the whiskers marks 1.5IQR out from each side of the box. The data points in this range outside the box, is also known as the lower and upper suspected outliers of the sample. The short perpendicular line at the end of each whisker, is also known as the lower and upper fence. The data points outside the lower and upper fence are denoted as outliers, usually marked as small dots or circles, like in a scatterplot. The underlying sample of data points of a boxplot might follow a host of different distributions, or none at all. Such distributions might be the Normal, Laplace, or Cauchy. The IQR may already be further defined in such a case. E.g. when a sample has a Normal distribution, the IQR becomes equivalent to 1.35 times the standard deviation, $1.35\sigma$, due to equation 28. [32] The median becomes equivalent to the mean. In the type of boxplot defined by the R function `graphics::boxplot`, 99.3 % of the data points are expected to reside between the lower and upper fence (including the box) due to equation 29, and 0.698 % of the data points are expected to be outliers because of equation 30.

$$2\Phi^{-1}(0.75)\sigma \approx 1.35\sigma \tag{28}$$

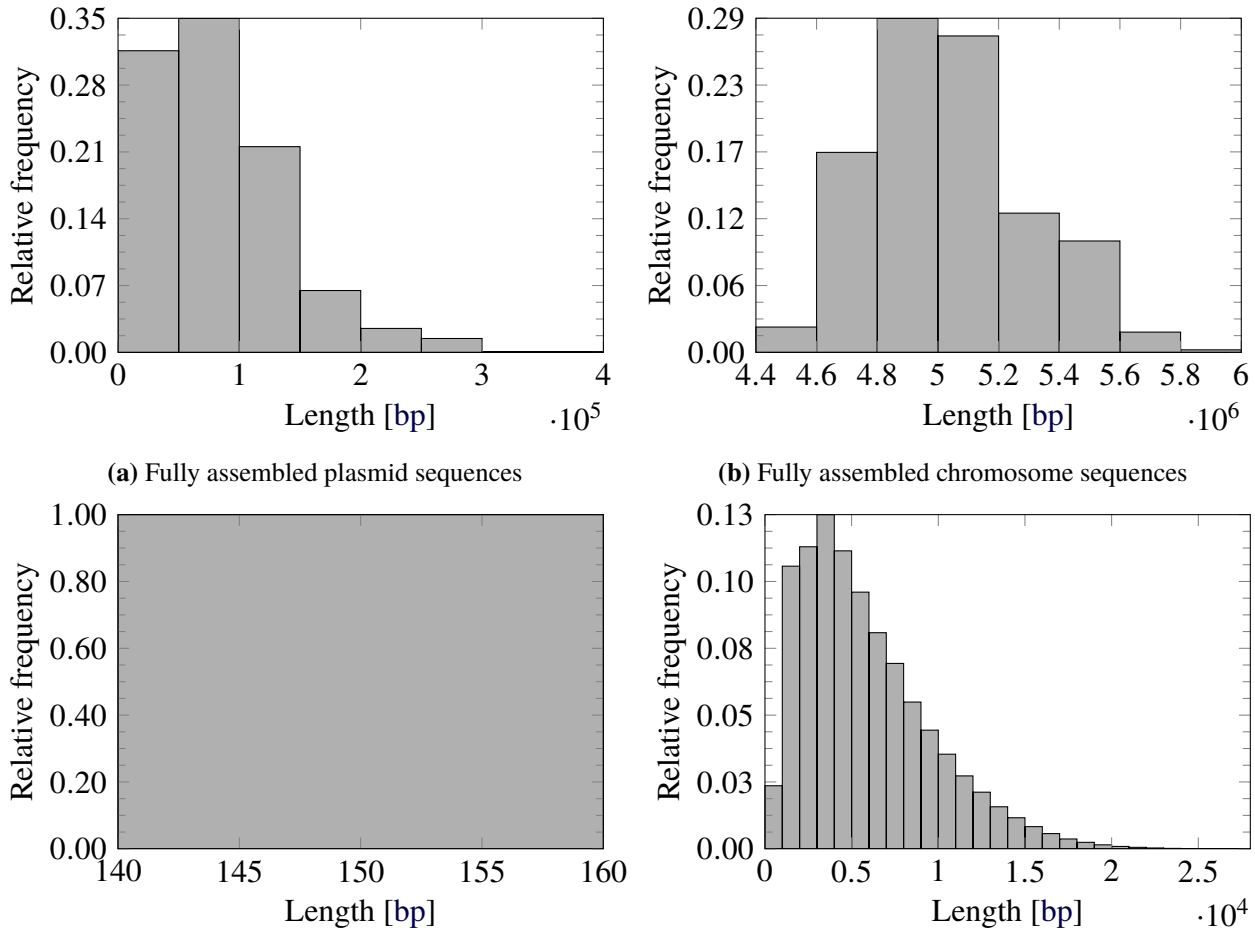
$$2\Phi\left(\Phi^{-1}(0.75)\cdot(2\cdot 1.5+1)\right)-1 = 0.99302 = a \tag{29}$$

$$1-a = 0.00698 \tag{30}$$

# 3 Results

## 3.1 Length distribution histograms

Figure 11 shows histograms of how the sequence lengths, measured in base pairs (bp), are distributed across the four sequence types analyzed in the text: whole plasmid sequences (11a), whole chromosomal sequences (11b), *in silico* simulated Illumina (ART) sequences (11c), and *in silico* simulated Nanopore (DeepSimulator) sequences (11d).




**(a)** Fully assembled plasmid sequences

**(b)** Fully assembled chromosome sequences

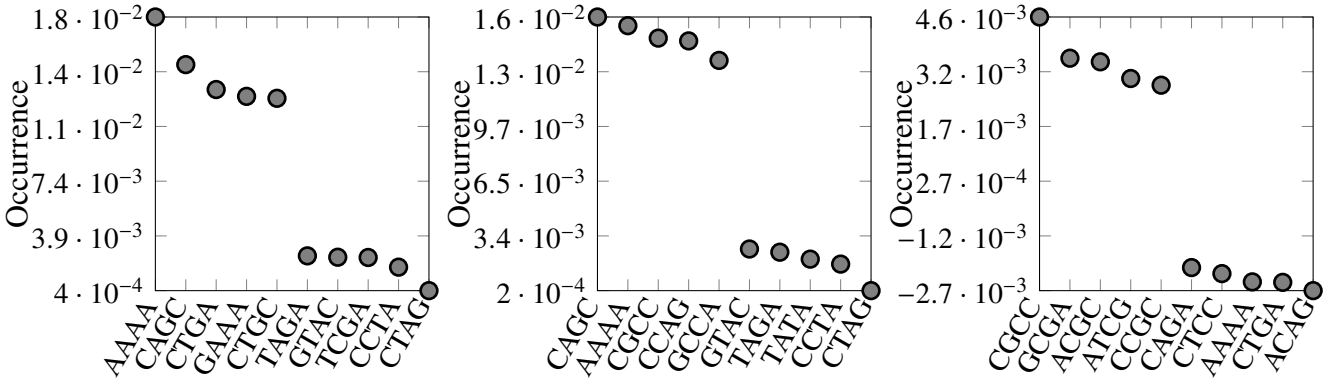
**(c)** Simulated ART sequences

**(d)** Simulated NanoPore sequences

**Figure 11:** Length distribution histograms of the four main types of sequences used in the classification pipeline.

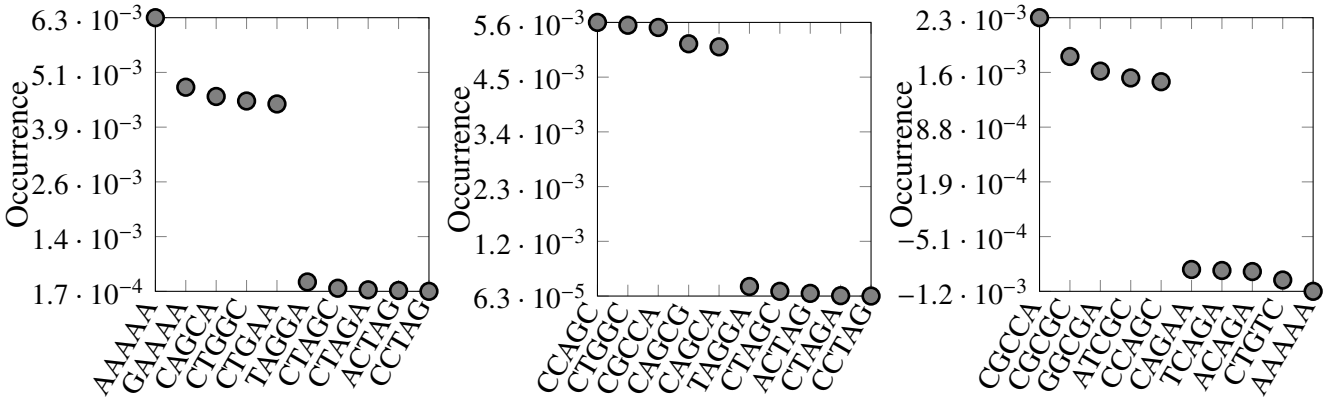
## 3.2 Expected K-mer occurrences

Figure 12 and 13 shows the first and last five data points in three distributions of decreasing occurrence for canonical 4- and 5-mers, respectively, i.e. the **(a)** subplots show the occurrence of plasmid K-mers, the **(b)** subplots show the occurrence of chromosomal K-mers, and the **(c)** subplots show the difference in occurrence of K-mers from chromosomes to plasmids.




**(a)** `plasmids` class **(b)** `chromosomes` class **(c)** Difference from `chromosomes` to `plasmids` class

**Figure 12:** First and last five data points in the distribution of occurrence of canonical 4-mers in the training data.




**(a)** `plasmids` class **(b)** `chromosomes` class **(c)** Difference from `chromosomes` to `plasmids` class

**Figure 13:** First and last five data points in the distribution of occurrence of canonical 5-mers in the training data.

## 3.3 ANOVA on the initial genome-wise cross-validated training data classification using the classification accuracy as the response

Before we try to classify reads, it is important to first investigate if it is at all possible to discriminate chromosomes from plasmids, based solely on K-mer statistics. Thus, we start out by classifying full genome sequences in a way similar to that we use for the reads later.

A genome-wide cross-validation was performed, i.e. each genome (chromosome + plasmids) were used as test set, and all other as training set. For each test set a range of methods were tested. In table 3 and figure 14 on page 21 is displayed the classification accuracy for the various methods, after

cross-validation, presented as the coefficient summary of a ANOVA analysis. Linear model formula of the intercept in table 3 on page 21 is displayed in equation 31.

$$\text{Accuracy} \sim \text{K(4)} + \text{Method(9NN.chebyshev)} + \text{Canonical(No)} \tag{31}$$

**Table 3:** Anova coefficient summary for the fully assembled (FA) genome-wise cross-validated training data classification using accuracy as the response, sorted in decreasing order in terms of the estimated coefficient column, `Estimate`.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | 1 | $2.1 \cdot 10^{-3}$ | 476.25 | 0 |
| K5 | $-1.87 \cdot 10^{-4}$ | $1.72 \cdot 10^{-3}$ | −0.11 | 0.91 |
| K3 | $-1.87 \cdot 10^{-4}$ | $1.72 \cdot 10^{-3}$ | −0.11 | 0.91 |
| K6 | $-4.67 \cdot 10^{-4}$ | $1.72 \cdot 10^{-3}$ | −0.27 | 0.79 |
| Method3NN.euclidean | $-6.72 \cdot 10^{-4}$ | $2.43 \cdot 10^{-3}$ | −0.28 | 0.78 |
| Method9NN.euclidean | $-7.47 \cdot 10^{-4}$ | $2.43 \cdot 10^{-3}$ | −0.31 | 0.76 |
| Method3NN.chebyshev | $-8.97 \cdot 10^{-4}$ | $2.43 \cdot 10^{-3}$ | −0.37 | 0.71 |
| Method3NN.manhattan | $-8.97 \cdot 10^{-4}$ | $2.43 \cdot 10^{-3}$ | −0.37 | 0.71 |
| Method3NN.minkowski | $-8.97 \cdot 10^{-4}$ | $2.43 \cdot 10^{-3}$ | −0.37 | 0.71 |
| K2 | $-9.9 \cdot 10^{-4}$ | $1.72 \cdot 10^{-3}$ | −0.58 | 0.56 |
| Method3NN.BinaryLogRatio | $-1.12 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −0.46 | 0.64 |
| Method9NN.minkowski | $-1.31 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −0.54 | 0.59 |
| Method9NN.manhattan | $-1.31 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −0.54 | 0.59 |
| K7 | $-1.42 \cdot 10^{-3}$ | $1.72 \cdot 10^{-3}$ | −0.83 | 0.41 |
| Method9NN.BinaryLogRatio | $-1.79 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −0.74 | 0.46 |
| Method1NN.euclidean | $-3.06 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −1.26 | 0.21 |
| Method1NN.chebyshev | $-3.14 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −1.29 | 0.2 |
| CanonicalYes | $-3.2 \cdot 10^{-3}$ | $8.58 \cdot 10^{-4}$ | −3.73 | $2.42 \cdot 10^{-4}$ |
| Method1NN.minkowski | $-3.29 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −1.35 | 0.18 |
| Method1NN.manhattan | $-3.29 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −1.35 | 0.18 |
| Method1NN.BinaryLogRatio | $-3.29 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −1.35 | 0.18 |
| MethodRandomForest | $-3.55 \cdot 10^{-3}$ | $2.43 \cdot 10^{-3}$ | −1.46 | 0.14 |
| K8 | $-4.97 \cdot 10^{-3}$ | $1.72 \cdot 10^{-3}$ | −2.9 | $4.15 \cdot 10^{-3}$ |
| K1 | $-8.11 \cdot 10^{-2}$ | $1.72 \cdot 10^{-3}$ | −47.28 | $4.41 \cdot 10^{-121}$ |

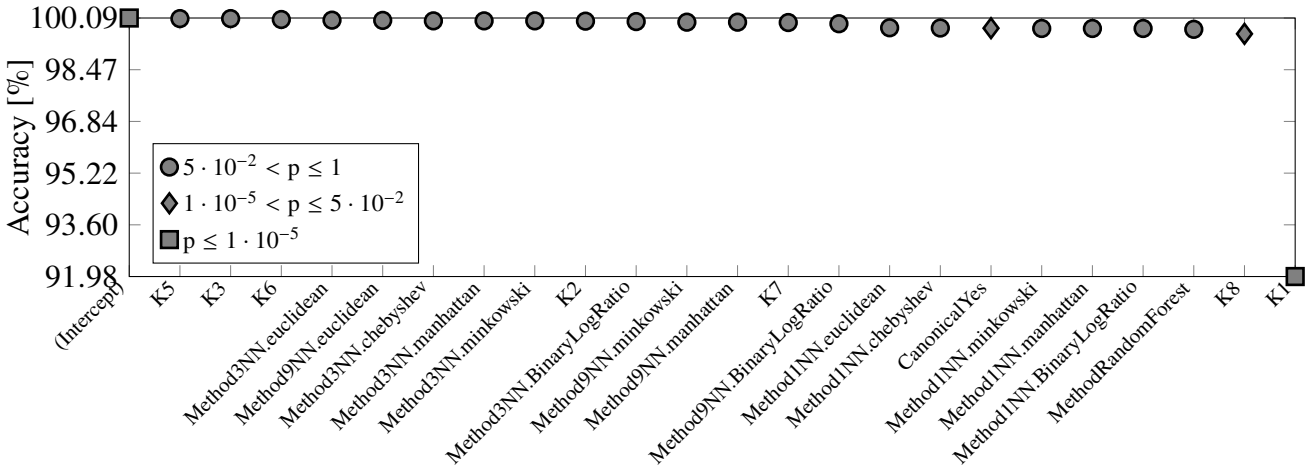



**Figure 14:** Classification accuracy for the various methods, after cross validation. The coefficient estimates of table 3 are plotted against the row names, with three levels of p-value as meta, as shown in the legend.

## 3.4 The *in silico* simulated Illumina read sequence data classification results

### 3.4.1 ANOVA on the Illumina test data and classifier parameter combinations using the binary classification class label accuracy as the response

The short paired-end reads from Illumina sequencing is what we meet in most practical cases. We wanted to test if it is possible to discriminate read-pairs coming from a plasmid from those coming from a chromosome.

Again, we used a genome-wise cross-validation, where each test set consisted of 50 % chromosome based read-pairs, and 50 % plasmid based read-pairs, a maximum of 500 read-pairs from the chromosome, and 500 read-pairs from the plasmid for a genome with one sufficiently long plasmid. The training set was the full length sequences, as before.

In table 4 and figure 15 on page 23, is displayed the classification accuracy for the various methods, we show how accuracies vary by the various method combinations in the ANOVA analysis of the accuracy of the binary classification on the Illumina sequences. Linear model formula of the intercept in table 4 on page 23 is displayed in equation 32.

$$\text{Accuracy} \sim \text{K(5)} + \text{Method(9NN.minkowski)} + \text{Canonical(Yes)} + \text{Err(No)} \tag{32}$$

**Table 4:** Anova coefficient summary for simulated Illumina reads, sorted in decreasing order in terms of the estimated coefficient column, `Estimate`.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | 0.61 | $7.59 \cdot 10^{-3}$ | 80.18 | $3.66 \cdot 10^{-138}$ |
| Method9NN.manhattan | $2.94 \cdot 10^{-17}$ | $9.6 \cdot 10^{-3}$ | $3.07 \cdot 10^{-15}$ | 1 |
| ErrYes | $-6.11 \cdot 10^{-4}$ | $3.39 \cdot 10^{-3}$ | −0.18 | 0.86 |
| Method3NN.minkowski | $-7.59 \cdot 10^{-3}$ | $9.6 \cdot 10^{-3}$ | −0.79 | 0.43 |
| Method3NN.manhattan | $-7.59 \cdot 10^{-3}$ | $9.6 \cdot 10^{-3}$ | −0.79 | 0.43 |
| Method1NN.manhattan | $-8.99 \cdot 10^{-3}$ | $9.6 \cdot 10^{-3}$ | −0.94 | 0.35 |
| Method1NN.minkowski | $-8.99 \cdot 10^{-3}$ | $9.6 \cdot 10^{-3}$ | −0.94 | 0.35 |
| CanonicalNo | $-9.71 \cdot 10^{-3}$ | $3.39 \cdot 10^{-3}$ | −2.86 | $4.75 \cdot 10^{-3}$ |
| Method9NN.euclidean | $-2.43 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −2.53 | $1.21 \cdot 10^{-2}$ |
| Method1NN.euclidean | $-2.77 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −2.88 | $4.45 \cdot 10^{-3}$ |
| Method3NN.euclidean | $-2.95 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −3.07 | $2.5 \cdot 10^{-3}$ |
| Method9NN.BinaryLogRatio | $-2.97 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −3.09 | $2.3 \cdot 10^{-3}$ |
| K2 | $-3.15 \cdot 10^{-2}$ | $4.16 \cdot 10^{-3}$ | −7.57 | $2.13 \cdot 10^{-12}$ |
| Method3NN.BinaryLogRatio | $-3.58 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −3.73 | $2.59 \cdot 10^{-4}$ |
| Method1NN.BinaryLogRatio | $-3.63 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −3.78 | $2.19 \cdot 10^{-4}$ |
| Method1NN.chebyshev | $-3.92 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −4.09 | $6.7 \cdot 10^{-5}$ |
| Method9NN.chebyshev | $-4.01 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −4.18 | $4.67 \cdot 10^{-5}$ |
| Method3NN.chebyshev | $-4.29 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −4.47 | $1.44 \cdot 10^{-5}$ |
| MethodRandomForest | $-5.13 \cdot 10^{-2}$ | $9.6 \cdot 10^{-3}$ | −5.34 | $2.87 \cdot 10^{-7}$ |
| K8 | $-7.9 \cdot 10^{-2}$ | $4.16 \cdot 10^{-3}$ | −19.01 | $3.88 \cdot 10^{-44}$ |

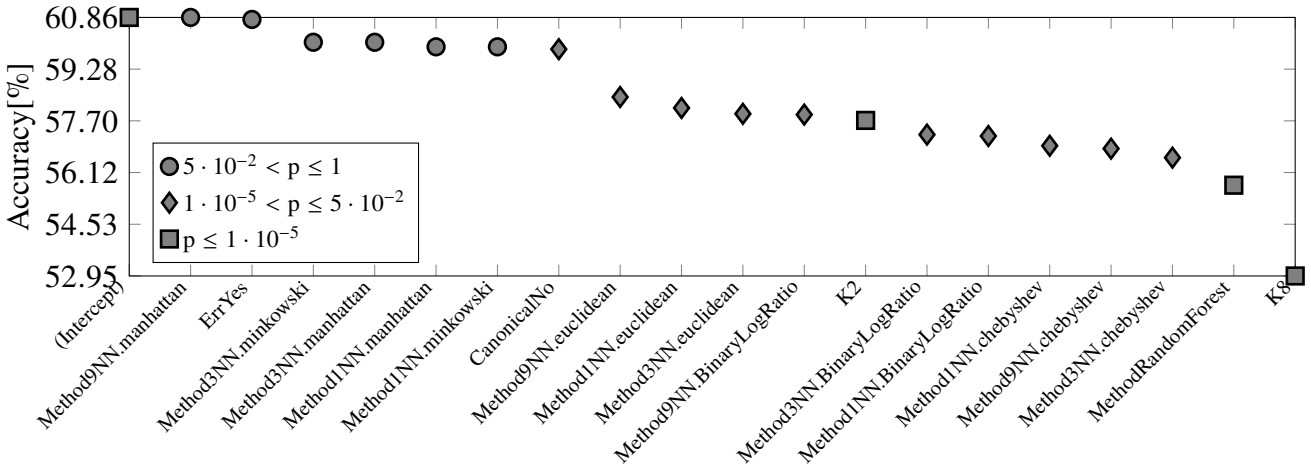



**Figure 15:** How accuracies vary by the various method combinations in classifying the *in silico* simulated Illumina reads. The coefficient estimates of table 4 are plotted against the row names, with three levels of p-value as meta, as shown in the legend.

### 3.4.2 Stacked data forming the basis of the intercept combination in the ANOVA Illumina accuracy analysis

Since the accuracy is now somewhat decreased from its earlier levels, we want to understand more why we miss so many chromosomes or plasmids. In table 5 we have listed, for each of the methods in table 4 with p-value greater than 0.05, the True Positive, False Positives, True Negatives, and False Negatives, along with precision, sensitivity class, but the opposite choice is equally relevant.

**Table 5:** Classification summary given the intercept as well as the two next lines in terms of coefficient estimate as the table stands in the Illumina accuracy ANOVA analysis in table 4 on page 23. *K* indicates K-mer length. *Method* indicates classification method. *Err* indicates presence/absence of sequencing error. *C* indicates if the K-mers are canonical or not. *TP* is the number of true positives. *FP* is the number of false positives. *TN* is the number of true negatives. *FN* is the number of false negatives. *Acc* is the classification accuracy. *TPR* is the sensitivity. *TNR* is the specificity. *PPV* is the precision.

| K | Method | Err | C | TP | FP | TN | FN | Acc | TPR | TNR | PPV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 1NN.minkowski | No | Yes | 206611 | 151253 | 59351 | 3993 | 0.63 | 0.98 | 0.28 | 0.58 |
| 5 | 1NN.manhattan | No | Yes | 206611 | 151253 | 59351 | 3993 | 0.63 | 0.98 | 0.28 | 0.58 |
| 5 | 3NN.minkowski | No | Yes | 208045 | 148703 | 61901 | 2559 | 0.64 | 0.99 | 0.29 | 0.58 |
| 5 | 3NN.manhattan | No | Yes | 208045 | 148703 | 61901 | 2559 | 0.64 | 0.99 | 0.29 | 0.58 |
| 5 | 9NN.minkowski | No | Yes | 207538 | 142256 | 68348 | 3066 | 0.65 | 0.99 | 0.32 | 0.59 |
| 5 | 9NN.manhattan | No | Yes | 207538 | 142256 | 68348 | 3066 | 0.65 | 0.99 | 0.32 | 0.59 |
| 5 | 1NN.minkowski | Yes | Yes | 210991 | 155346 | 59521 | 3876 | 0.63 | 0.98 | 0.28 | 0.58 |
| 5 | 1NN.manhattan | Yes | Yes | 210991 | 155346 | 59521 | 3876 | 0.63 | 0.98 | 0.28 | 0.58 |
| 5 | 3NN.minkowski | Yes | Yes | 212302 | 152550 | 62317 | 2565 | 0.64 | 0.99 | 0.29 | 0.58 |
| 5 | 3NN.manhattan | Yes | Yes | 212302 | 152550 | 62317 | 2565 | 0.64 | 0.99 | 0.29 | 0.58 |
| 5 | 9NN.minkowski | Yes | Yes | 211775 | 145749 | 69118 | 3092 | 0.65 | 0.99 | 0.32 | 0.59 |
| 5 | 9NN.manhattan | Yes | Yes | 211775 | 145749 | 69118 | 3092 | 0.65 | 0.99 | 0.32 | 0.59 |

## 3.5 The *in silico* simulated Nanopore read sequence data classification results

We repeated the cross-validation exercise again for the Oxford Nanopore sequences, again using the full genome sequences for training and a set of 500 reads from both chromosomes and plasmids as test set for each genome.

### 3.5.1 ANOVA on the Nanopore test data and classifier parameter combinations using the binary classification class label accuracy as the response

In table 6 and figure 16 on page 25, is displayed the classification accuracy for the various methods, we show how accuracies vary by the various method combinations in the ANOVA analysis of the accuracy of the binary classification on the Nanopore sequences. Linear model formula of the intercept in table 6 on page 25 is displayed in equation 33.

$$\text{Accuracy} \sim \text{K(5)} + \text{Method(3NN.euclidean)} + \text{Canonical(Yes)} \tag{33}$$

**Table 6:** Anova coefficient summary for simulated Nanopore reads, sorted in decreasing order in terms of the estimated coefficient column, `Estimate`.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | 0.87 | $2.21 \cdot 10^{-2}$ | 39.46 | $7.68 \cdot 10^{-53}$ |
| Method9NN.euclidean | $-3.95 \cdot 10^{-3}$ | $2.86 \cdot 10^{-2}$ | −0.14 | 0.89 |
| Method1NN.euclidean | $-9.09 \cdot 10^{-3}$ | $2.86 \cdot 10^{-2}$ | −0.32 | 0.75 |
| Method3NN.minkowski | $-1.2 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.42 | 0.68 |
| Method3NN.manhattan | $-1.2 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.42 | 0.68 |
| Method9NN.manhattan | $-1.69 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.59 | 0.56 |
| Method9NN.minkowski | $-1.69 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.59 | 0.56 |
| Method3NN.BinaryLogRatio | $-1.75 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.61 | 0.54 |
| CanonicalNo | $-1.91 \cdot 10^{-2}$ | $1.01 \cdot 10^{-2}$ | −1.88 | $6.33 \cdot 10^{-2}$ |
| Method1NN.manhattan | $-1.95 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.68 | 0.5 |
| Method1NN.minkowski | $-1.95 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.68 | 0.5 |
| Method9NN.BinaryLogRatio | $-2.32 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.81 | 0.42 |
| Method1NN.BinaryLogRatio | $-2.42 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −0.84 | 0.4 |
| Method3NN.chebyshev | $-7.14 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −2.49 | $1.48 \cdot 10^{-2}$ |
| Method1NN.chebyshev | $-7.44 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −2.6 | $1.12 \cdot 10^{-2}$ |
| K2 | $-7.67 \cdot 10^{-2}$ | $1.24 \cdot 10^{-2}$ | −6.18 | $2.82 \cdot 10^{-8}$ |
| Method9NN.chebyshev | $-7.73 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −2.7 | $8.54 \cdot 10^{-3}$ |
| MethodRandomForest | $-9.19 \cdot 10^{-2}$ | $2.86 \cdot 10^{-2}$ | −3.21 | $1.95 \cdot 10^{-3}$ |
| K8 | −0.31 | $1.24 \cdot 10^{-2}$ | −24.97 | $1.15 \cdot 10^{-38}$ |

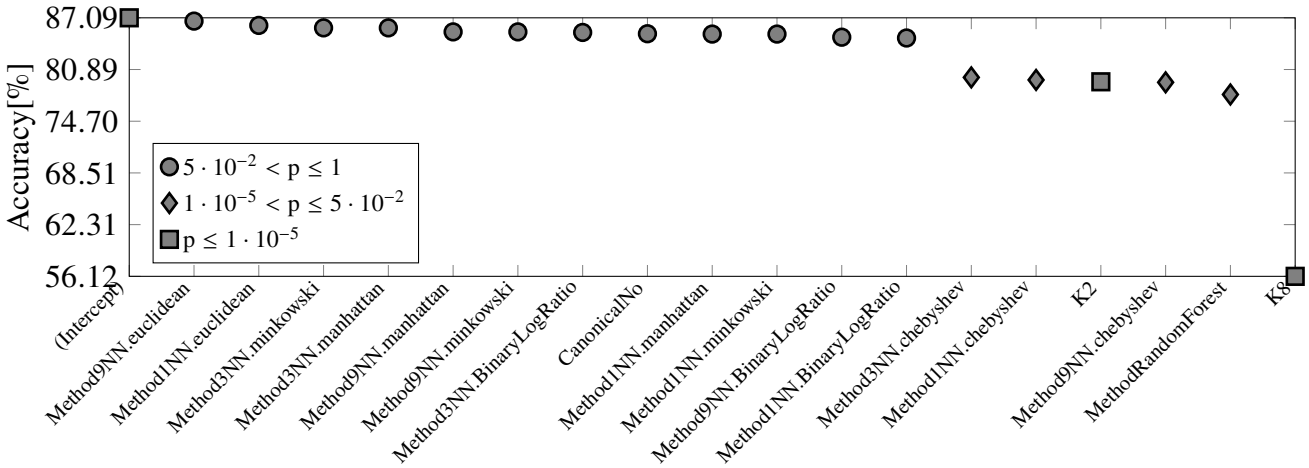



**Figure 16:** How accuracies vary by the various method combinations in classifying the *in silico* simulated Nanopore reads. The coefficient estimates of table 6 are plotted against the row names, with three levels of p-value as meta, as shown in the legend.

### 3.5.2 Stacked data forming the basis of the intercept combination in the ANOVA Nanopore accuracy analysis

Table 7 lists the data forming the basis for the combinations in table 6 with a p-value greater than 0.05.

**Table 7:** Classification summary given the intercept as well as the two next lines in terms of coefficient estimate as the table stands in the Nanopore accuracy ANOVA analysis in table 6 on page 25. *K* indicates K-mer length. *Method* indicates classification method. *C* indicates if the K-mers are canonical or not. *TP* is the number of true positives. *FP* is the number of false positives. *TN* is the number of true negatives. *FN* is the number of false negatives. *Acc* is the classification accuracy. *TPR* is the sensitivity. *TNR* is the specificity. *PPV* is the precision.

| K | Method | C | TP | FP | TN | FN | Acc | TPR | TNR | PPV |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 1NN.minkowski | Yes | 113817 | 25790 | 91513 | 3486 | 0.88 | 0.97 | 0.78 | 0.82 |
| 5 | 1NN.euclidean | Yes | 114251 | 28251 | 89052 | 3052 | 0.87 | 0.97 | 0.76 | 0.80 |
| 5 | 1NN.manhattan | Yes | 113817 | 25790 | 91513 | 3486 | 0.88 | 0.97 | 0.78 | 0.82 |
| 5 | 3NN.minkowski | Yes | 114109 | 23219 | 94084 | 3194 | 0.89 | 0.97 | 0.80 | 0.83 |
| 5 | 3NN.euclidean | Yes | 114572 | 25079 | 92224 | 2731 | 0.88 | 0.98 | 0.79 | 0.82 |
| 5 | 3NN.manhattan | Yes | 114109 | 23219 | 94084 | 3194 | 0.89 | 0.97 | 0.80 | 0.83 |
| 5 | 9NN.minkowski | Yes | 113384 | 22195 | 95108 | 3919 | 0.89 | 0.97 | 0.81 | 0.84 |
| 5 | 9NN.euclidean | Yes | 114063 | 24103 | 93200 | 3240 | 0.88 | 0.97 | 0.79 | 0.83 |
| 5 | 9NN.manhattan | Yes | 113384 | 22195 | 95108 | 3919 | 0.89 | 0.97 | 0.81 | 0.84 |
| 5 | 1NN.BinaryLogRatio | Yes | 113757 | 24507 | 92796 | 3546 | 0.88 | 0.97 | 0.79 | 0.82 |
| 5 | 3NN.BinaryLogRatio | Yes | 113789 | 22654 | 94649 | 3514 | 0.89 | 0.97 | 0.81 | 0.83 |
| 5 | 9NN.BinaryLogRatio | Yes | 113058 | 21593 | 95710 | 4245 | 0.89 | 0.96 | 0.82 | 0.84 |
| 5 | 1NN.minkowski | No | 113005 | 25132 | 92171 | 4298 | 0.87 | 0.96 | 0.79 | 0.82 |
| 5 | 1NN.euclidean | No | 112978 | 27789 | 89514 | 4325 | 0.86 | 0.96 | 0.76 | 0.80 |
| 5 | 1NN.manhattan | No | 113005 | 25132 | 92171 | 4298 | 0.87 | 0.96 | 0.79 | 0.82 |
| 5 | 3NN.minkowski | No | 113751 | 24224 | 93079 | 3552 | 0.88 | 0.97 | 0.79 | 0.82 |
| 5 | 3NN.euclidean | No | 114088 | 26771 | 90532 | 3215 | 0.87 | 0.97 | 0.77 | 0.81 |
| 5 | 3NN.manhattan | No | 113751 | 24224 | 93079 | 3552 | 0.88 | 0.97 | 0.79 | 0.82 |
| 5 | 9NN.minkowski | No | 113169 | 23190 | 94113 | 4134 | 0.88 | 0.96 | 0.80 | 0.83 |
| 5 | 9NN.euclidean | No | 113603 | 25343 | 91960 | 3700 | 0.88 | 0.97 | 0.78 | 0.82 |
| 5 | 9NN.manhattan | No | 113169 | 23190 | 94113 | 4134 | 0.88 | 0.96 | 0.80 | 0.83 |
| 5 | 1NN.BinaryLogRatio | No | 113677 | 26868 | 90435 | 3626 | 0.87 | 0.97 | 0.77 | 0.81 |
| 5 | 3NN.BinaryLogRatio | No | 114100 | 25602 | 91701 | 3203 | 0.88 | 0.97 | 0.78 | 0.82 |
| 5 | 9NN.BinaryLogRatio | No | 113593 | 24420 | 92883 | 3710 | 0.88 | 0.97 | 0.79 | 0.82 |

## 3.6 Principal component analysis on the K-mer occurrences of the training data of the parameters partly forming the basis of the intercepts in all three ANOVA analyses

The unique combinations of K (K-mer length) and C (canonical/non-canonical) from table 5 and 7 on page 24 and 26 are simply canonical and non-canonical 5-mers. Omitting the non-canonical bit, letting K have a standard deviation of 1, yields canonical 4-, 5-, and 6-mers. Principal component analysis (PCA), using the R function `stats::prcomp`, were performed on the training data for these three K-mer combinations. The two classes within the first two principal components were plotted in

light grey lower case letter p's (plasmid class) and black uppercase K's (chromosome class) in figure 17, 18, and 19 on page 27 to 28. Additionally, in figure 18, the classes of the Illumina (ART, figure 18b)[12] and Nanopore (DS, figure 18c) test data, were plotted in cyan (Illumina plasmid class), red (Illumina chromosome class), orange (Nanopore plasmid class), and blue (Nanopore chromosome class). In both the Illumina and Nanopore case, the dimensions of each data point are the sample mean drawn from the same dimension in the underlying 500 test data points. For the Illumina case, this causes the number of K-mers to not always to be $2(150\,\text{bp} - 5 + 1) = 292$, cf. equation 2.

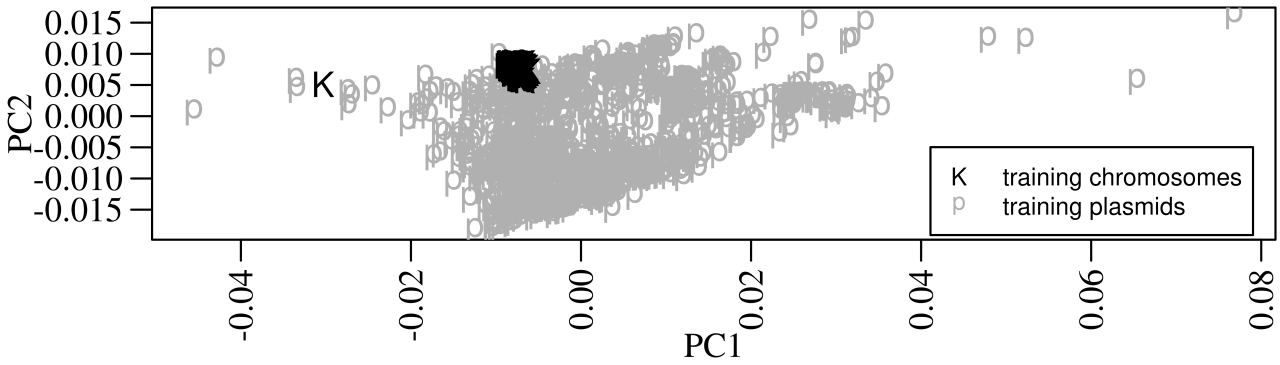
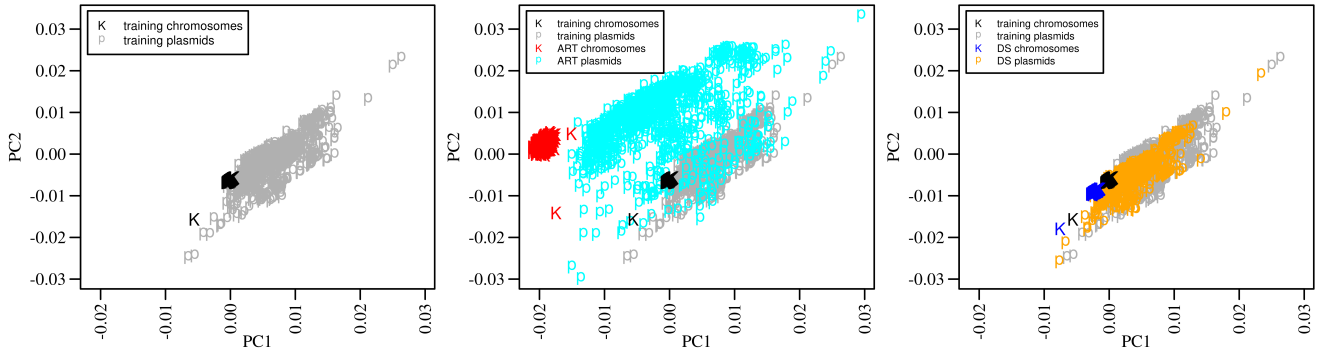



**Figure 17:** PCA plot of the plasmid and chromosome class for the first two components of a principal component analysis of the canonical canonical 4-mer occurrence training dataset. The data points for the chromosomal and plasmid classes are marked with black, uppercase K's, and light grey, lowercase p's, respectively.




**(a)** Training **(b)** Training & Illumina **(c)** Training & Nanopore

**Figure 18:** PCA plot of the plasmid and chromosome class for the first two components of a principal component analysis of the canonical 5-mer occurrence training, Illumina, and Nanopore datasets. The data points for the chromosomal and plasmid classes are marked with black, uppercase K's, and light grey, lowercase p's, respectively. Additionally, marked in red, cyan, blue, and orange letters in the same way, respectively, are plasmid and chromosomal 5-mer occurrence data points of the Illumina (ART) and Nanopore (DS, DeepSimulator) test data, summarized irrespectively of the boolean sequencing error level (error-free and non-error-free profiles are summed) so as to be comparable to the training data.

[12] summarized irrespective of the boolean sequencing error level: error-free and non-error-free occurrences are summed, a new occurrence profile are calculated from there

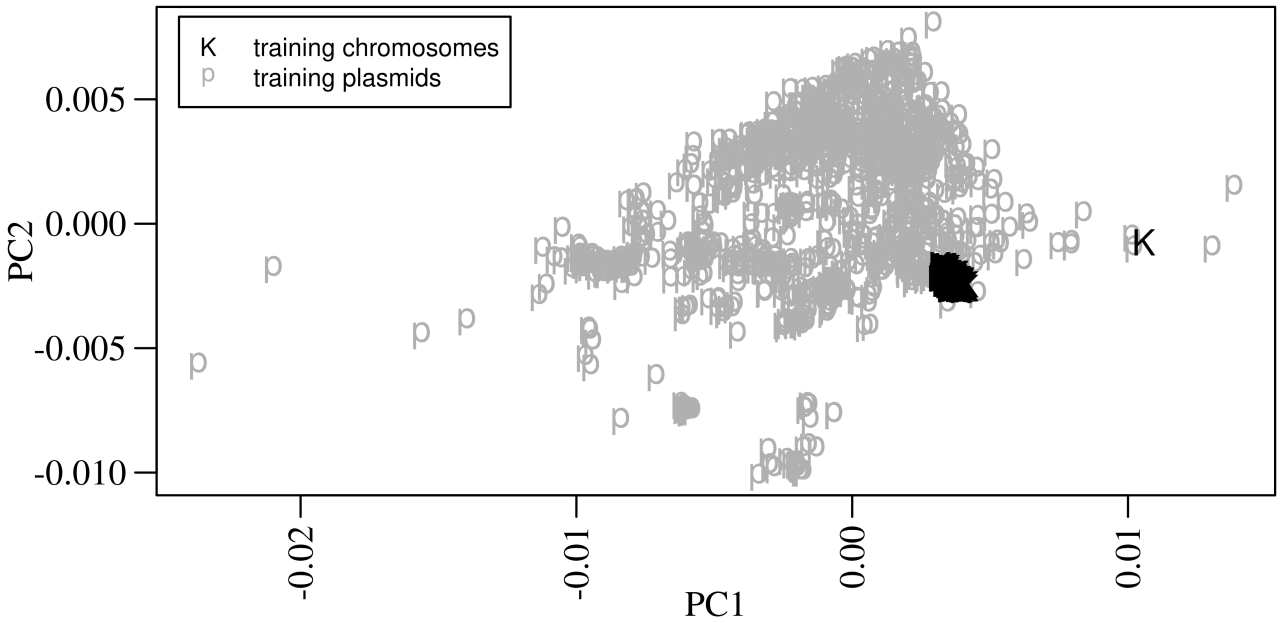



**Figure 19:** PCA plot of the plasmid and chromosome class for the first two components of a principal component analysis of the canonical canonical 6-mer occurrence training dataset. The data points for the chromosomal and plasmid classes are marked with black, uppercase K's, and light grey, lowercase p's, respectively.

## 3.7 Prediction probability boxplot

Figure 20 on page 29 presents boxplots of the prediction probability for the positive class (P), the plasmid class, of the data forming the basis for the intercept in the ANOVA analyses with accuracy as the response in table 3 (FA), 4 (ART), and 6 (DS) respectively on pages 21, 23, and 25. The prediction probabililty of the positive class is an output from a fitted classifier, as described in section 2.3.5 on page 13, and can be interpreted as the "probability of plasmid" for the intercepts, i.e. for the fully assembled case (FA), equation 31 on page 21 signifies K = 4, Method = 9NN.chebyshev, and Canonical = No; for the Illumina case (ART), equation 32 on page 22 signifies K = 5, Method = 9NN.minkowski, Canonical = Yes, and Err = No; for the Nanopore case (DS), equation 33 on page 24 signifies K = 5, Method = 3NN.euclidean, and Canonical = Yes; table 8 presents metrics relevant to the boxplot in figure 20, on a per-box-basis, such as standard deviation and IQR.

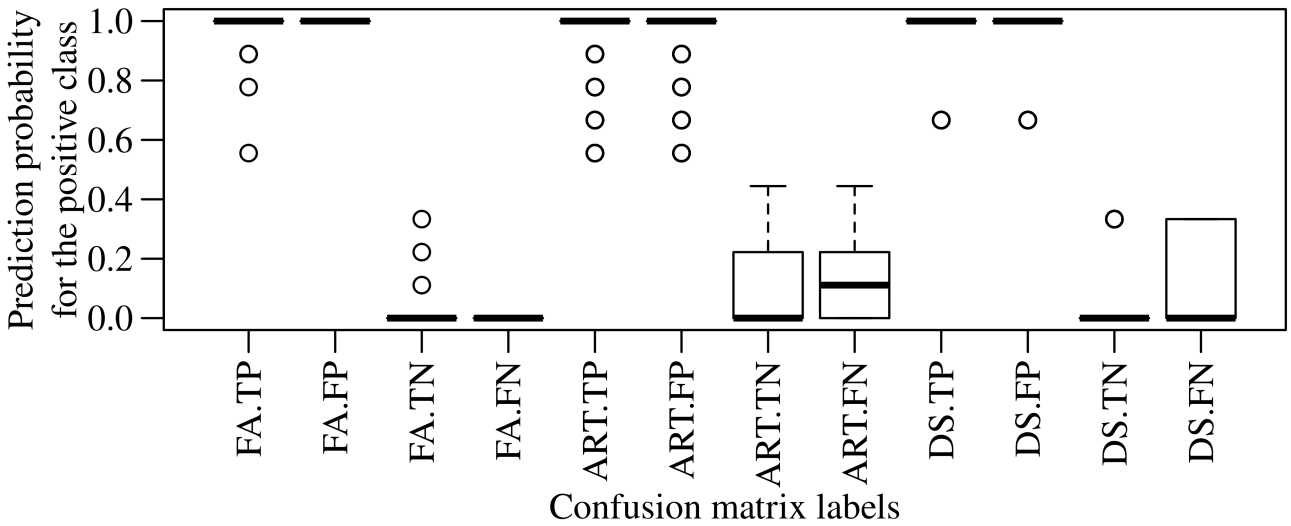



**Figure 20:** Prediction probability boxplot for the terms in the confusion matrix of the accuracy intercepts in the ANOVA analyses of table 3, 4, and 6 respectively on pages 21, 23, and 25. The x-axis of the box plot shows the combination of the four binary classification confusion matrix terms, true positive (TP), false positive (FP), true negative (TN), and false negative (FN), and the three classes of sequence length classified, fully assembled (FA), simulated Illumina reads (ART), and simulated Nanopore reads (DS). The y-axis of the box plot shows the prediction probability, `predict_proba`, of the positive class (P), the plasmid class, and since only two classes were classified, the prediction probability of the negative class (N, the chromosomes class) is `predict_proba(N)` = 1 − `predict_proba(P)`.

**Table 8:** Prediction probability (for the positive class, P) summary for the accuracy intercept in table 3, 4, and 6 respectively on pages 21, 23, and 25. *Column name descriptions:* Task indicates either FA (fully assembled sequences), ART (`art_Illumina` based sequences), or DS (DeepSimulator based sequences). Label indicated either TP (true positive), FP (false positive), TN (true negative), or FN (false negative). x indicates the sample mean. s indicates the sample deviation. n indicates number of observations. Columns $Q_{\varphi}$ where $\varphi[\%] \in \{0, 25, 50, 75, 100\}$ indicates percentiles. IQR indicates the interquartile range, $Q_{75\,\%} - Q_{25\,\%}$. LS indicates lower suspected outliers, $Q_{25\,\%} - 1.5 \cdot \text{IQR}$. US indicates lower suspected outliers, $Q_{75\,\%} + 1.5 \cdot \text{IQR}$. LO indicates lower outliers, $Q_{25\,\%} - 3 \cdot \text{IQR}$. UO indicates upper outliers, $Q_{75\,\%} + 3 \cdot \text{IQR}$.

| Task | Label | x | s | n | $Q_{0\,\%}$ | $Q_{25\,\%}$ | $Q_{50\,\%}$ | $Q_{75\,\%}$ | $Q_{100\,\%}$ | IQR | LS | US | LO | UO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FA | TP | 1 | 0.02 | 1218 | 0.56 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| FA | FP | 1 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| FA | TN | 0 | 0.02 | 452 | 0 | 0 | 0 | 0 | 0.33 | 0 | 0 | 0 | 0 | 0 |
| FA | FN | 0 | N/A | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ART | TP | 0.99 | 0.04 | 23984 | 0.56 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| ART | FP | 0.95 | 0.11 | 16326 | 0.56 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| ART | TN | 0.11 | 0.14 | 8054 | 0 | 0 | 0 | 0.22 | 0.44 | 0.22 | N/A | 0.56 | N/A | 0.89 |
| ART | FN | 0.16 | 0.16 | 396 | 0 | 0 | 0.11 | 0.22 | 0.44 | 0.22 | N/A | 0.56 | N/A | 0.89 |
| DS | TP | 0.99 | 0.06 | 24319 | 0.67 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| DS | FP | 0.95 | 0.12 | 5067 | 0.67 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| DS | TN | 0.03 | 0.09 | 19870 | 0 | 0 | 0 | 0 | 0.33 | 0 | 0 | 0 | 0 | 0 |
| DS | FN | 0.11 | 0.16 | 618 | 0 | 0 | 0 | 0.33 | 0.33 | 0.33 | N/A | 0.83 | N/A | N/A |

# 4 Discussion

## 4.1 Length distribution histograms

The subfigures of figure 11 on page 19 indicates that fully assembled plasmids are roughly (81 ± 59) kb long, fully assembled chromosomes being (5.0 ± 0.3) Mb long, *in silico* simulated Illumina Hiseq 2500 reads being 150 bp long, and *in silico* simulated Oxford Nanopore reads being (5.8 ± 3.7) kb long. This means that the fully assembled chromosomes are 60X, 33531X, and 861X greater in length compared to the fully assembled plasmids, ART reads, and Nanopore reads respectively. The fully assembled plasmids are 2 % shorter than the chromosomes, and 540X and 12X greater in length compared to the ART and Nanopore reads respectively. The Nanopore sequences are 37X greater in length compared to the Illumina sequences.

Considering the four sequence lengths presented in figure 11 on page 19, and the fact that there exists 1024 unique 5-mers (table 1), equation 2 on page 7 will then yield 4900, 79, 0.29, 5.6 expected 5-mers per unique 5-mer for the fully assembled chromosomal sequences, fully assembled plasmid sequences, simulated Illumina reads (paired-end), and simulated Nanopore reads case respectively. E.g. for the plasmid case, expected sequence length established earlier is 81 kb, which yields $81 \times 10^3 - 5 + 1 \approx 81 \times 10^3$ 5-mers. Every unique 5-mer is thus expected to be found $81 \times 10^3 / 80994 = 79$ times along any one of the fully assembled plasmid sequences.

Later in the text, the implications thus far drawn from section 3.1, to the result sections 3.4 and 3.5 will be further discussed. It can be said that when comparing a longer sequence to a shorter one, the deeper K-mer profile established from the longer sequence make sequence differences more evident compared to a profile of lesser depth derived from a shorter sequence. A 37X increase in sequence length[13] leads to a 42 % increase in accuracy[14].

## 4.2 Expected K-mer occurrences

The difference in occurrence subplots, figure 12c and 13c on page 20, shows that the plasmid inducing K-mers has lower %GC content compared to their chromosomal counterparts. E.g. the plasmid inducing 4-mer AAAA has 0 %GC, while the chromosomal inducing 5-mer CGCGC has 100 %GC. Thus, it might seem GC-content becomes the most important property for separating plasmids from chromosomes. However, in figure 12c and 13c we see that the 5-mers with the largest difference in occurrence (the chromosome associated 4- and 5-mers) between chromosome and plasmids also include some with a mix of AT and GC, e.g. ATCGC. Thus, a pure GC content classifier will probably work to some degree, but will not utilize all discriminative information in K-mers. That said, K-mer occurrence, e.g. GC-content, remains independent of classification methods for predicting plasmid and chromosomal sequences.

[13] from 150 b to 5.8 kb
[14] from 61 % to 87 %

## 4.3 ANOVA on the initial genome-wise cross-validated training data classification (the fully assembled sequences (FA) case) using the classification accuracy as the response

In the previous section, only K-mer occurrences in the training data were considered. This indicates there are some differences in composition between chromosomes and plasmids, but in order to see how well the sequences can be classified, some classifications methods based on K-mer data needs to be implemented. K-mer occurrences were used as predictors in a range of machine learning methods to see if they could predict the correct class[15] for a number of test-set cases. For all selected methods, all combinations of K-values[16] and canonical[17] were tried, see equation 31 and table 3 on page 21.

The results of the ANOVA analysis of the accuracies of the binary classification of the fully assembled plasmid and chromosome sequences in table 3 and figure 14 on page 21, show extremely good accuracy, very close to 100 % for almost all methods, no one method dropping substantially below the others—even RandomForest and the KNN methods with K = 1. The plot along with table 3 shows how both the Canonical factor, as well as K-mer lengths 8 and 1, can be rejected, due to p-values less than 0.05. The best method combination (K-mer length, classifier, canonical) is listed in equation 31, but as we can see from figure 14, virtually all methods perform extremely well for these data.

This result illustrates that the use of K-mer statistics extracts the information required to discriminate plasmids from chromosomes in *E.coli*. It is, however, a "best case scenario," in the sense that the data is full length genome sequences, generating a huge number of K-mers, and with no sequencing errors. But, at least this tells us that K-mer methods have the potential we need.

## 4.4 The *in silico* simulated Illumina read sequence data classification results

### 4.4.1 ANOVA on the Illumina test data and classifier parameter combinations using the binary classification class label accuracy as the response

In table 4 and figure 15 on page 23, we show how accuracies vary by the various method combinations. There are two main differences from the previous section: (1) accuracies are now much lower, and (2) there are larger differences between the method combinations. The best accuracy is just over 60 %. Since we classify the exact same number of chromosome and plasmid read-pairs each time, a pure guessing would result in around 50 % accuracy. Thus, none of the method combinations are able to extract very much information from these short reads.

In equation 32 on page 22 is listed the best method combination, K = 5, Method = 9NN.minkowski, Canonical = Yes, and Err = No, and in table 4 is the result of an ANOVA, indicating which other method combinations are performing equally well based on their cross-validated accuracy. The best accuracy combination in equation 32 was used as a reference combination (Intercept), and all other

[15] chromosome or plasmid, plasmid being the positive class
[16] K-mer lengths, from 1 to 8
[17] canonical or non-canonical

combinations are tested against this, the results displayed in table 4. The six first combinations do not produce significantly poorer accuracy than the best. These include variations on the KNN algorithm with one, three, and nine neighbours, either the Manhattan or the Minkowski distance metric, in addition to toggling the sequence simulation error profile to the *On* state (Err = Yes).

### 4.4.2 Stacked data forming the basis of the intercept combination in the ANOVA Illumina accuracy analysis

In this analysis we used equal proportions (1:1) of plasmid (positive class) and chromosome reads as test sets during cross validation. This causes the number of True Positives and True Negatives to remain comparable. From table 5, we observe that the number of TP is much greater than TN, and the results are similar for all models listed. Thus, all models have the same tendency to classify reads as plasmids, also when they are from chromosomes, i.e. the False Positive case.

## 4.5 The *in silico* simulated Nanopore read sequence data classification results

### 4.5.1 ANOVA on the Nanopore test data and classifier parameter combinations using the binary classification class label accuracy as the response

In table 6 and figure 16 on page 25, we show how accuracies vary by the various method combinations. There are two main differences from the previous two sections: (1) accuracies are on an in-between level from the previous two sections, much higher than the Illumina case at least, (2) differences between the method combinations have become more evened out, almost like the fully assembled sequence case (first section).

The best accuracy is almost 90 %. Like in the previous section, we classify the exact same number of chromosome and plasmid read-pairs each time, thus a pure guessing would result in around 50 % accuracy. The method combinations seems to be able to extract a lot more information from these longer reads, and also seem to be equally well-suited for the task.

In equation 33 on page 24 is listed the best method combination, K = 5, Method = 3NN.euclidean, and Canonical = Yes, and in table 6 is the result of an ANOVA, indicating which other method combinations are performing equally well based on their cross-validated accuracy. The best accuracy combination in equation 33 was used as a reference combination (Intercept), and all other combinations are tested against this, the results displayed in table 6. The six first combinations do not produce significantly poorer accuracy than the best. These include some, but not all of, the variations of the KNN algorithm using one, three, and nine neighbours and the Euclidean (equation 13 on page 15), Minkowski (equation 11 on page 14), Manhattan (equation 12 on page 15), or the binary log-ratio (equation 15 on page 15) distance metric. Using canonical K-mers (Canonical = No) is also a valid combinational choice, with the added benefit of saving computational runtime.

### 4.5.2 Stacked data forming the basis of the intercept combination in the ANOVA Nanopore accuracy analysis

Overall, the binary classification accuracy was much greater in the Nanopore case (equation 6 on page 25) compared to that of the Illumina case (table 4 on page 23). This is due to the number of falsely classified plasmids (FP) decreasing from the Illumina case (table 5) to the Nanopore case table 7, leading to an increase in both precision and selectivity. The binary log ratio distance metric (equation 15 on page 15) for nine neighbours yields the lowest number of false positives, thus being the combination leading to the highest selectivity.

## 4.6 Principal component analysis on the K-mer occurrences of the training data of the parameters partly forming the basis of the intercepts in all three ANOVA analyses

Figure 17, 18, and 19 on pages 27 to 28, shows that the PC1 : PC2 ratio decreases as K increases. The chromosomal data points marked as black uppercase letter K's, are gathered close together, while the plasmids marked with grey lowercase letter p's are spread out around them[18], and also under (in the same coordinates as) the chromosomal grouping. For figure 18b the Illumina (ART) plasmid data points are much more spread out compared to the training plasmids. The Illumina chromosomes has greater volatility (standard deviation) compared to the training chromosomes, and the two groups also seems to be quite some distance away from each other. This increased variance might explain the high number of false positives for the Illumina case, in that underlying data cannot provide enough information to predict the chromosomal class. For the Nanopore (DS, figure 18c) case, the opposite seems to occur, i.e. the orange coloured Nanopore plasmid are positioned well inside the training plasmid grouping, while the blue Nanopore chromosomal grouping only has a smaller bias to the training chromosomes compared to the Illumina case in figure 18b. We also observe a bias, i.e. the Illumina read data are re-located in comparison to the training data, and this might contribute to more classification errors.

Upon further inspection of the data points underlying the plots, the single chromosomal outlier in all five[19] plots has an accession number of `LR025099.1`, a chromosome discovered by Aitor Gonzaga Moltó at the German Collection of Microorganisms and Cell Cultures GmbH (DSMZ) on 2018-07-11 [33]. It looks like this is either not an *E.coli* chromosome at all, or may be misassembled, perhaps due to a mixup with plasmid reads.

## 4.7 Prediction probability boxplot

figure 20 and table 8 on page 29 shows that for Illumina data, the prediction probability can tell us something about certainty, but only when predicting the chromosomal class. When the prediction is plasmid, both true positives and false positive have very high probability of plasmid; but when the

[18] mostly to one side
[19] figure 17, 18a, 18b, 18c, and 19

prediction is chromosome, the true negatives tend to have lesser prediction probability for the positive class (probability of plasmid) than the false negatives. In the latter case the median is at zero, while in the FN case the interquartile range is almost 0.22, while the standard deviation is 0.16, indicating the underlying FN data points are close to being Normally distributed. For the Nanopore data the picture is similar, but less pronounced. The results for the Full Assembly data are less informative, since the number of misclassified sequences are extremely low, both ways.

# 5 Conclusion

The aim of the project was to investigate to what extent some machine learning methods are able to classify chromosomal reads from plasmid reads based on K-mer statistics. It has been shown that K-mer methods work extremely well for fully assembled genome sequences. For the *in silico* simulated Illumina *HiSeq 2500* 150 bp reads, the binary classification accuracy drops to just above 60 %. For the *in silico* simulated Nanopore $(5.8 \pm 3.7)$ kb reads, the binary classification accuracy retains a far better level right above 90 %. Falsely classified reads mainly gets classified as plasmids (false positives). Two reasons for the positive class to mainly be the one being misclassified, might be:

(i) Chromosomes are quite similar to each other.

(ii) Plasmids are quite different from each other.

In terms of the biology of chromosomes and plasmids, the chromosomes are "the genomes," already containing vital genes. Chromosomal sequences has little room for difference between each other. All variants of *E. coli* has to have a minimum of genetic machinery in order to be an *E. coli*. Although this limits how much chromosomal sequences can vary, *E. coli* varies more than most bacteria. On the other hand, plasmids are additional, non-vital sequences. Plasmids are thus able to vary greatly in contents without harming the *E. coli*, thus plasmid derived K-mer occurrences also are able to vary grately, cf. the K-mer occurrence PCA plots on figure 17, 18, and 19 on pages 27 to 28. Figure 18 shows Illumina plasmid reads might be overfitted due to the data point grouping having high bias and high variance, cf. figure 10, while the Illumina chromosomes seems to be underfitted, at least compared to the Nanopore chromosomes in figure 18c. Underfitted data cannot provide enough information to predict the correct class. The underfitted Illumina chromosomes might thus explain the greater number of false positives in the Illumina case compared to the Nanopore case[20].

# 6 Further work

In a few years time, as more sequencing labs are able to get hold of third generation sequencers, longer reads, such as Nanopore reads, will become common sightings in the lab. Thus, further work on classifying these longer types of reads makes sense.

Using more complex "black-box" classifiers, such as prototype methods and adaptive nearest-neighbour methods, e.g. the *discriminant adaptive nearest-neighbor* [21, 34] might increase the

[20] cf. table 5 and 7 on pages 24 and 26

accuracy a few percentage points. Classification using partial least squares classification methods, such as those provided by the R package `pls` [35], could also be investigated.

Classifying other types of plasmid data, such as linear plasmids, low quality assemblies, and contigs as per Krawczyk *et al.* [11], might also be explored.

# References

[1] Weichun Huang, Leping Li, Jason R. Myers, & Gabor T. Marth (12 2012), "ART: a next-generation sequencing read simulator," Bioinformatics, vol. 28, pp. 593–594. doi:10.1093/bioinformatics/btr708.

[2] Yu Li, Renmin Han, Chongwei Bi, Mo Li, Sheng Wang, and Xin Gao (2018), "DeepSimulator: a deep simulator for Nanopore sequencing," Bioinformatics, vol. 34, no. 17, pp. 2899–2908. doi:10.1093/bioinformatics/bty223.

[3] Stephen Baker, Nicholas Thomson, François-Xavier Weill, and Kathryn E Holt (2018), "Genomic insights into the emergence and spread of antimicrobial-resistant bacterial pathogens," Science, vol. 360, no. 6390, pp. 733–738.

[4] Mohammad Shah Jalal, Md Zohorul Islam, Avijit Dutta, Pangkaj Kumar Dhar, Avijit Das, Mohammad Mahbub Hasan, Himel Barua, Paritosh Kumar Biswas, and Abdul Ahad (2018), "Antibiotic resistant zoonotic bacteria in Irrawaddy squirrel (Callosciurus pygerythrus)," Veterinary medicine and science.

[5] Nenad Macesic, Fernanda Polubriaginof, and Nicholas P Tatonetti (2017), "Machine learning: novel bioinformatics approaches for combating antimicrobial resistance," Current opinion in infectious diseases, vol. 30, no. 6, pp. 511–517.

[6] Per Kristian Knudsen, Karianne Wiger Gammelsrud, Kristian Alfsnes, Martin Steinbakk, Tore G. Abrahamsen, Fredrik Müller, & Jon Bohlin (12 2017), "Transfer of a $bla_{\text{CTX-M-1}}$-carrying plasmid between different Escherichia coli strains within the human gut explored by whole genome sequencing analyses," Scientific Reports, vol. 8. doi:10.1038/s41598-017-18659-2.

[7] Sara Goodwin, John D. McPherson, & W. Richard McCombie (6 2016), "Coming of age: ten years of next-generation sequencing technologies," Nature Reviews, vol. 17, pp. 333–351.

[8] Eric S. Lander & Michael S. Waterman (1988), "Genomic mapping by fingerprinting random clones: a mathematical analysis," Genomics, vol. 2, no. 3, pp. 231–239.

[9] Erika Check Hayden (2014), "Is the $1,000 genome for real?," Nature News.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015), "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," Software available from tensorflow.org.

[11] Pawel S. Krawczyk, Leszek Lipinski, & Andrzej Dziembowski (2018), "PlasFlow: predicting plasmid sequences in metagenomic data using genome signatures," Nucleic Acids Research, vol. 46, no. 6.

[12] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull, "Graphviz and dynagraph – static and dynamic graph drawing tools," in GRAPH DRAWING SOFTWARE, pp. 127–148, Springer-Verlag (2003).

[13] Lars Snipen, & Kristian Hovde Liland (3 2015), "micropan: an R-package for microbial pan-genomics," BMC Bioinformatics, vol. 79. doi:10.1186/s12859-015-0517-0.

[14] GNU Project, "GNU C Library." https://www.gnu.org/software/libc/. Retrieved on 2019-01-14.

[15] Rich Felker et al., "musl libc." https://www.musl-libc.org/. Retrieved on 2019-01-14.

[16] "FASTQ Format Specification." http://maq.sourceforge.net/fastq.shtml. Retrieved on 2019-01-14.

[17] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup (2009), "The Sequence Alignment/Map format and SAMtools," Bioinformatics, vol. 25, no. 16, pp. 2078–2079.

[18] Dmitry Antipov, Nolan Hartwick, Max Shen, Mikhail Raiko, Alla Lapidus, & Pavel A. Pevzner (4 2016), "plasmidSPAdes: Assembling Plasmids from Whole Genome Sequencing Data," bioRxiv. doi:10.1101/048942.

[19] Illumina (December 2014), "Estimating Sequencing Coverage." https://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf. Retrieved on 2018-12-09.

[20] Robert V. Hogg, Joseph W. McKean, & Allen T. Craig (2013), Introduction to Mathematical Statistics, 7 ed. Pearson. ISBN 13: 978-0-321-79543-4.

[21] T. Hastie, R. Tibshirani, and J.H. Friedman (2009), The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer series in statistics, Springer.

[22] Gareth James, Daniela Witten, Trevor Hastie, & Robert Tibshirani (2013), An Introduction to Statistical Learning with Applications in R. New York: Springer Science+Business Media. doi:10.1007/978-1-4614-7138-7.

[23] Fengfeng Zhou and Ying Xu (2010), "cBar: a computer program to distinguish plasmid-derived from chromosome-derived sequence fragments in metagenomics data," Bioinformatics, vol. 26, no. 16, pp. 2051–2052.

[24] Alessandra Carattoli, Ea Zankari, Aurora García-Fernández, Mette Voldby Larsen, Ole Lund, Laura Villa, Frank Møller Aarestrup, and Henrik Hasman (2014), "In silico detection and typing of plasmids using PlasmidFinder and plasmid multilocus sequence typing," Antimicrobial agents and chemotherapy, vol. 58, no. 7, pp. 3895–3903.

[25] Roye Rozov, Aya Brown Kav, David Bogumil, Naama Shterzer, Eran Halperin, Itzhak Mizrahi, and Ron Shamir (2017), "Recycler: an algorithm for detecting plasmids from de novo assembly graphs," Bioinformatics, vol. 33, no. 4, pp. 475–482.

[26] Leo Breiman (2001), "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32.

[27] Leo Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone (1984), "Classification and regression trees. Wadsworth and Brooks," Cole Statistics/Probability Series.

[28] Corrado Gini (1912), Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche. Studi economico-giuridici pubblicati per cura della facoltà di Giurisprudenza della R. Università di Cagliari, Tipogr. di P. Cuppini.

[29] Gilles Louppe (2014), "Understanding random forests: From theory to practice," arXiv preprint arXiv:1407.7502.

[30] Claude E Shannon and Warren Weaver (1949), "The mathematical theory of information,"

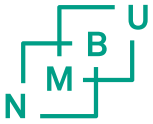
[31] Charles D Ghilani (2010), Adjustment computations: spatial data analysis. John Wiley & Sons.

[32] Tom W. Kirkman, "Box Plot: Display of Distribution." http://www.physics.csbsju.edu/stats/box2.html. Retrieved on 2019-06-17.

[33] Aitor Gonzaga Moltó, "Direct Submission. Submitted (11-JUL-2018) Leibniz-Institut DSMZ-Deutsche Sammlung von Mikro, Mikrobielle Genomforschung, Inhoffenstrasse 7 B, 38124, Germany." https://www.ncbi.nlm.nih.gov/nuccore/LR025099.1. Retrieved on 2019-07-05.

[34] Trevor Hastie and Robert Tibshirani, "Discriminant adaptive nearest neighbor classification and regression," in Advances in Neural Information Processing Systems, pp. 409–415 (1996).

[35] Bjørn-Helge Mevik, Ron Wehrens, and Kristian Hovde Liland (2019), pls: Partial Least Squares and Principal Component Regression. R package version 2.7-1.




**Norges miljø- og biovitenskapelige universitet**
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway