

Architecture of large projects in bioinformatics (ADP)

Lecture 09

Łukasz P. Kozłowski

Warsaw, 2025

Reproducible research

When you present the data use **both**:

Plots

Tables

When you present the data use **both**:

Plots

the best solution, very natural and easy to interpretation

Tables

When you present the data use **both**:

Plots

the best solution, very natural and easy to interpretation
(but also prone for miss-interpretation)

Tables

When you present the data use **both**:

Plots

the best solution, very natural and easy to interpretation
(but also prone for miss-interpretation)

Tables

harder to interpret in short time, but higher information content

When you present the data use **both**:

Plots

the best solution, very natural and easy to interpretation
(but also prone for miss-interpretation)

Tables

harder to interpret in short time, but higher information content

* Raw data

for the sake of completeness if you can add them

* The scripts

for the sake of reproducibility if you can add them

When you present the data use **both**:

Plots

the best solution, very natural and easy to interpretation
(but also prone for miss-interpretation)

Tables

harder to interpret in short time, but higher information content

* Raw data

for the sake of completeness if you can add them

* The scripts

for the sake of reproducibility if you can add them

* The text

When you present the data use **both**:

Plots

the best solution, very natural and easy to interpretation
(but also prone for miss-interpretation)

Tables

harder to interpret in short time, but higher information content

**Reproducible
research**

* Raw data

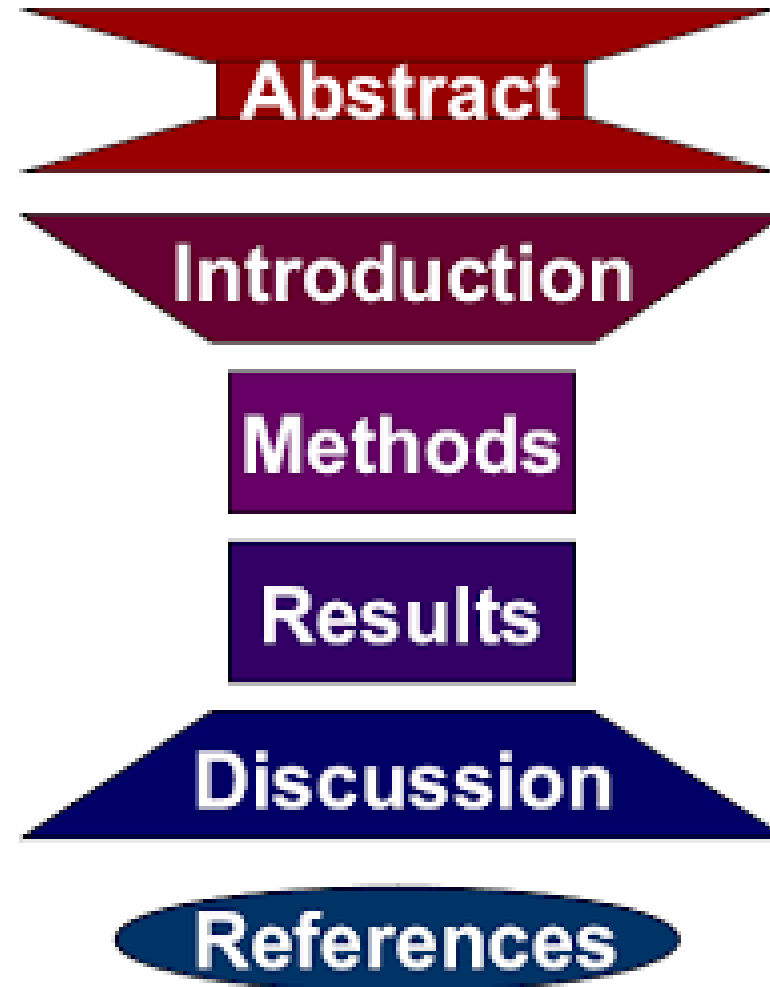
for the sake of completeness if you can add them

* The scripts

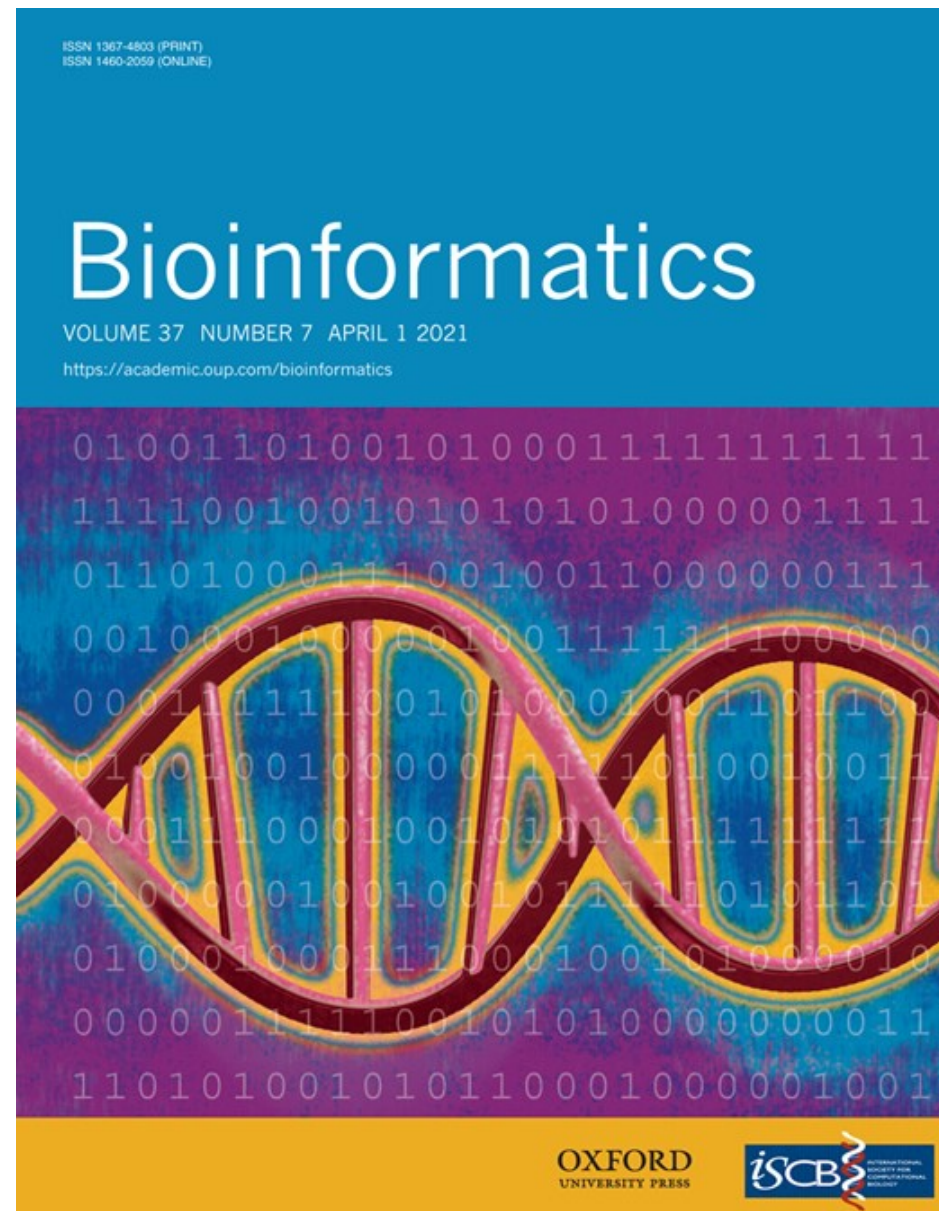
for the sake of reproducibility if you can add them

* The text


Scientific article structure



Scientific article structure



Sequence analysis

CARE: context-aware sequencing read error correction**Felix Kallenborn** *, **Andreas Hildebrandt** and **Bertil Schmidt***

Department of Computer Science, Johannes Gutenberg University, Mainz 55122, Germany

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on May 8, 2020; revised on July 14, 2020; editorial decision on August 6, 2020; accepted on August 14, 2020

Abstract

Motivation: Error correction is a fundamental pre-processing step in many Next-Generation Sequencing (NGS) pipelines, in particular for *de novo* genome assembly. However, existing error correction methods either suffer from high false-positive rates since they break reads into independent *k*-mers or do not scale efficiently to large amounts of sequencing reads and complex genomes.

Results: We present CARE—an alignment-based scalable error correction algorithm for Illumina data using the concept of minhashing. Minhashing allows for efficient similarity search within large sequencing read collections which enables fast computation of high-quality multiple alignments. Sequencing errors are corrected by detailed inspection of the corresponding alignments. Our performance evaluation shows that CARE generates significantly fewer false-positive corrections than state-of-the-art tools (Musket, SGA, BFC, Lighter, Bcool, Karect) while maintaining a competitive number of true positives. When used prior to assembly it can achieve superior *de novo* assembly results for a number of real datasets. CARE is also the first multiple sequence alignment-based error corrector that is able to process a human genome Illumina NGS dataset in only 4 h on a single workstation using GPU acceleration.

Availability and implementation: CARE is open-source software written in C++ (CPU version) and in CUDA/C++ (GPU version). It is licensed under GPLv3 and can be downloaded at <https://github.com/fkallen/CARE>.

Contact: kallenborn@uni-mainz.de or bertil.schmidt@uni-mainz.de,


Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

The progress of Next-Generation Sequencing (NGS) technologies has a major impact on biological and medical research, as illustrated by the role of sequencing in understanding the origins and epidemiology

of the given collection of sequencing reads. The goal is to determine *k*-mers which are error-free (*trusted*) or erroneous (*untrusted*) with high probability. Often, this is achieved by choosing an abundance threshold, assuming that an

Sequence analysis

CARE: context-aware sequencing read error correctionFelix Kallenborn *, Andreas Hildebrandt and Bertil Schmidt*

Department of Computer Science, Johannes Gutenberg University, Mainz 55122, Germany

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on May 8, 2020; revised on July 14, 2020; editorial decision on August 6, 2020; accepted on August 14, 2020

Abstract

Motivation: Error correction is a fundamental pre-processing step in many Next-Generation Sequencing (NGS) pipelines, in particular for *de novo* genome assembly. However, existing error correction methods either suffer from high false-positive rates since they break reads into independent *k*-mers or do not scale efficiently to large amounts of sequencing reads and complex genomes.

Results: We present CARE—an alignment-based scalable error correction algorithm for Illumina data using the concept of minhashing. Minhashing allows for efficient similarity search within large sequencing read collections which enables fast computation of high-quality multiple alignments. Sequencing errors are corrected by detailed inspection of the corresponding alignments. Our performance evaluation shows that CARE generates significantly fewer false-positive corrections than state-of-the-art tools (Musket, SGA, BFC, Lighter, Bcool, Karect) while maintaining a competitive number of true positives. When used prior to assembly it can achieve superior *de novo* assembly results for a number of real datasets. CARE is also the first multiple sequence alignment-based error corrector that is able to process a human genome Illumina NGS dataset in only 4 h on a single workstation using GPU acceleration.

Availability and implementation: CARE is open-source software written in C++ (CPU version) and in CUDA/C++ (GPU version). It is licensed under GPLv3 and can be downloaded at <https://github.com/fkallen/CARE>.

Contact: kallenborn@uni-mainz.de or bertil.schmidt@uni-mainz.de,

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

The progress of Next-Generation Sequencing (NGS) technologies has a major impact on biological and medical research, as illustrated by the role of sequencing in understanding the origins and epidemiology

of the given collection of sequencing reads. The goal is to determine *k*-mers which are error-free (*trusted*) or erroneous (*untrusted*) with high probability. Often, this is achieved by choosing an abundance threshold, assuming that an

We present an MSA-based error corrector for Illumina data—called CARE—that can efficiently scale towards human-size genomes. CARE is based on a variant of *minhashing* to quickly find a set of candidate reads which are similar to a query read with high probability. This allows for fast construction of high-quality MSAs that are further refined and then inspected in detail to perform context-aware correction. We demonstrate that our approach is beneficial to

1. runtime and scalability,
2. reduction of false-positives corrections and
3. downstream analysis using *de novo* genome assembly

In particular, the speed of CARE is comparable to the fastest state-of-the-art k -mer spectrum methods but produces significantly fewer false-positive corrections while maintaining a competitive number of true positives. It efficiently scales towards large sequencing read collections (such as human genome NGS data) resulting in significantly shorter runtimes than the recent MSA-based methods while still producing around one order-of-magnitude fewer false positives on average. In particular, low numbers of false positives (i.e. fewer newly introduced errors) are an attractive feature for downstream analysis. We show that prior application of CARE results in superior *de novo* assembly results for several datasets compared to other error correction tools.

2 Approach

Consider a massive collection of sequencing reads. Our aim is to efficiently calculate similarities between them to construct meaningful MSAs. A brute-force approach that computes all pairwise alignments is computationally infeasible due to the large data sizes. MSA-based error correction approaches often consider *anchor* reads. Each anchor is aligned to a set of candidate reads to find a consensus sequence which is used for correction. A key aspect is the selection of candidate reads, which needs to be accurate and quick. For example, Coral (Salmela and Schröder, 2011) and ECHO (Kao et al., 2011) identify candidate reads that share at least one exact k -mer with the anchor. Unfortunately, this simple method has low sensitivity for large values of k , since many relevant candidate reads will be missed and has low selectivity for small values of k , since the number of identified candidate reads grows enormously.

We address this problem using a variant of *minhashing* to quickly find a set of candidate reads which are very similar to the anchor read with high probability. Read *signatures* are generated by applying a number of hash functions to each k -mer of a read and choosing the smallest hash value for each function. These signatures provide a memory-reduced representation which can be used for string comparison: the relative intersection ratios of two read signatures approximate the true Jaccard Index evaluated on their whole k -mer space. CARE is thus able to use a relatively small value of k ($k = 20$ by default) to identify candidate reads with both high selectivity and high sensitivity.

Our approach requires only a standard workstation with a sufficient amount of main memory. We take advantage of commonly available parallelism by exploiting multi-threading on modern multi-core CPUs. In addition, we provide a CUDA version to exploit GPUs for accelerating the compute-intensive parts of our pipeline.

3 Materials and methods

3.1 Algorithm

In the following, we explain the individual steps of the overall workflow of CARE (Fig. 1). In addition, there is an initial stage for database construction and a post-processing stage for consensus voting from different corrections of the same read.

3.1.1 Database construction

Consider a collection of reads $R = \{r_1, \dots, r_m\}$ and an input parameter k . Unless otherwise stated, we assume each read to be of equal length l . In a pre-processing step, all ambiguous nucleotides can be randomly replaced by A, C, G or T. This is required for reads to participate in error correction. Furthermore, let h_1, \dots, h_H be H hash functions that map k -mers to unique integers used for minhashing.

Each read $r_i \in R$ is transformed into its set of strand-neutral canonical k -mers (a canonical k -mer is the lexicographically smaller one of a k -mer and its reverse complement). Subsequently, each hash function h_j is applied to every canonical k -mer. Only the H smallest resulting integers for each hash function are kept representing the *signature* of r_i .

All signatures are inserted into a database. The database consists of H hash tables (key-value stores). The j th hash table is associated with the hash function h_j . In each bucket, we store the hash value as key and the IDs of all reads containing this hash value at position j of their signature. If a bucket exceeds a certain number of reads (default: $2.5 \times \text{estimated_coverage}$), it is considered uninformative and is removed from the hash table.

3.1.2 Database querying

We consider each $r_i \in R$ as anchor read. To find an initial set of candidate reads $C(r_i)$, the read signature of r_i is calculated again and used to query the database. The query returns a list of read IDs for each hash table j , i.e. these reads have the identical minhash signature at the position j . $C(r_i)$ is the union of the returned read IDs for all $j \in \{1, \dots, H\}$.

3.1.3 Filtering of candidate reads

Since minhashing is a probabilistic measure, it does not necessarily guarantee sufficiently high degrees of similarity between the anchor r_i and every read in $C(r_i)$. Thus, we further refine the set of candidates to obtain a potentially smaller set of highly similar candidates $F(r_i)$. Ideally, the filtered set $F(r_i)$ should only contain reads with high overlaps and very few mismatches to the anchor, thus maximizing the probability that they are sequenced from the same genomic

We present an MSA-based error corrector for Illumina data—called CARE—that can efficiently scale towards human-size genomes. CARE is based on a variant of *minhashing* to quickly find a set of candidate reads which are similar to a query read with high probability. This allows for fast construction of high-quality MSAs that are further refined and then inspected in detail to perform context-aware correction. We demonstrate that our approach is beneficial to

1. runtime and scalability,
2. reduction of false-positives corrections and
3. downstream analysis using *de novo* genome assembly

In particular, the speed of CARE is comparable to the fastest state-of-the-art k -mer spectrum methods but produces significantly fewer false-positive corrections while maintaining a competitive number of true positives. It efficiently scales towards large sequencing read collections (such as human genome NGS data) resulting in significantly shorter runtimes than the recent MSA-based methods while still producing around one order-of-magnitude fewer false positives on average. In particular, low numbers of false positives (i.e. fewer newly introduced errors) are an attractive feature for downstream analysis. We show that prior application of CARE results in superior *de novo* assembly results for several datasets compared to other error correction tools.

2 Approach

Consider a massive collection of sequencing reads. Our aim is to efficiently calculate similarities between them to construct meaningful MSAs. A brute-force approach that computes all pairwise alignments is computationally infeasible due to the large data sizes. MSA-based error correction approaches often consider *anchor* reads. Each anchor is aligned to a set of candidate reads to find a consensus sequence which is used for correction. A key aspect is the selection of candidate reads, which needs to be accurate and quick. For example, Coral (Salmela and Schröder, 2011) and ECHO (Kao et al., 2011) identify candidate reads that share at least one exact k -mer with the anchor. Unfortunately, this simple method has low sensitivity for large values of k , since many relevant candidate reads will be missed and has low selectivity for small values of k , since the number of identified candidate reads grows enormously.

We address this problem using a variant of *minhashing* to quickly find a set of candidate reads which are very similar to the anchor read with high probability. Read *signatures* are generated by applying a number of hash functions to each k -mer of a read and choosing the smallest hash value for each function. These signatures provide a memory-reduced representation which can be used for string comparison: the relative intersection ratios of two read signatures approximate the true Jaccard Index evaluated on their whole k -mer space. CARE is thus able to use a relatively small value of k ($k = 20$ by default) to identify candidate reads with both high selectivity and high sensitivity.

Our approach requires only a standard workstation with a sufficient amount of main memory. We take advantage of commonly available parallelism by exploiting multi-threading on modern multi-core CPUs. In addition, we provide a CUDA version to exploit GPUs for accelerating the compute-intensive parts of our pipeline.

3 Materials and methods

3.1 Algorithm

In the following, we explain the individual steps of the overall workflow of CARE (Fig. 1). In addition, there is an initial stage for database construction and a post-processing stage for consensus voting from different corrections of the same read.

3.1.1 Database construction

Consider a collection of reads $R = \{r_1, \dots, r_m\}$ and an input parameter k . Unless otherwise stated, we assume each read to be of equal length l . In a pre-processing step, all ambiguous nucleotides can be randomly replaced by A, C, G or T. This is required for reads to participate in error correction. Furthermore, let b_1, \dots, b_H be H hash functions that map k -mers to unique integers used for minhashing.

Each read $r_i \in R$ is transformed into its set of strand-neutral canonical k -mers (a canonical k -mer is the lexicographically smaller one of a k -mer and its reverse complement). Subsequently, each hash function b_j is applied to every canonical k -mer. Only the H smallest resulting integers for each hash function are kept representing the *signature* of r_i .

All signatures are inserted into a database. The database consists of H hash tables (key-value stores). The j th hash table is associated with the hash function b_j . In each bucket, we store the hash value as key and the IDs of all reads containing this hash value at position j of their signature. If a bucket exceeds a certain number of reads (default: $2.5 \times \text{estimated_coverage}$), it is considered uninformative and is removed from the hash table.

3.1.2 Database querying

We consider each $r_i \in R$ as anchor read. To find an initial set of candidate reads $C(r_i)$, the read signature of r_i is calculated again and used to query the database. The query returns a list of read IDs for each hash table j , i.e. these reads have the identical minhash signature at the position j . $C(r_i)$ is the union of the returned read IDs for all $j \in \{1, \dots, H\}$.

3.1.3 Filtering of candidate reads

Since minhashing is a probabilistic measure, it does not necessarily guarantee sufficiently high degrees of similarity between the anchor r_i and every read in $C(r_i)$. Thus, we further refine the set of candidates to obtain a potentially smaller set of highly similar candidates $F(r_i)$. Ideally, the filtered set $F(r_i)$ should only contain reads with high overlaps and very few mismatches to the anchor, thus maximizing the probability that they are sequenced from the same genomic

We present an MSA-based error corrector for Illumina data—called CARE—that can efficiently scale towards human-size genomes. CARE is based on a variant of *minhashing* to quickly find a set of candidate reads which are similar to a query read with high probability. This allows for fast construction of high-quality MSAs that are further refined and then inspected in detail to perform context-aware correction. We demonstrate that our approach is beneficial to

1. runtime and scalability,
2. reduction of false-positives corrections and
3. downstream analysis using *de novo* genome assembly

In particular, the speed of CARE is comparable to the fastest state-of-the-art k -mer spectrum methods but produces significantly fewer false-positive corrections while maintaining a competitive number of true positives. It efficiently scales towards large sequencing read collections (such as human genome NGS data) resulting in significantly shorter runtimes than the recent MSA-based methods while still producing around one order-of-magnitude fewer false positives on average. In particular, low numbers of false positives (i.e. fewer newly introduced errors) are an attractive feature for downstream analysis. We show that prior application of CARE results in superior *de novo* assembly results for several datasets compared to other error correction tools.

2 Approach

Consider a massive collection of sequencing reads. Our aim is to efficiently calculate similarities between them to construct meaningful MSAs. A brute-force approach that computes all pairwise alignments is computationally infeasible due to the large data sizes. MSA-based error correction approaches often consider *anchor* reads. Each anchor is aligned to a set of candidate reads to find a consensus sequence which is used for correction. A key aspect is the selection of candidate reads, which needs to be accurate and quick. For example, Coral (Salmela and Schröder, 2011) and ECHO (Kao et al., 2011) identify candidate reads that share at least one exact k -mer with the anchor. Unfortunately, this simple method has low sensitivity for large values of k , since many relevant candidate reads will be missed and has low selectivity for small values of k , since the number of identified candidate reads grows enormously.

We address this problem using a variant of *minhashing* to quickly find a set of candidate reads which are very similar to the anchor read with high probability. Read *signatures* are generated by applying a number of hash functions to each k -mer of a read and choosing the smallest hash value for each function. These signatures provide a memory-reduced representation which can be used for string comparison: the relative intersection ratios of two read signatures approximate the true Jaccard Index evaluated on their whole k -mer space. CARE is thus able to use a relatively small value of k ($k = 20$ by default) to identify candidate reads with both high selectivity and high sensitivity.

Our approach requires only a standard workstation with a sufficient amount of main memory. We take advantage of commonly available parallelism by exploiting multi-threading on modern multi-core CPUs. In addition, we provide a CUDA version to exploit GPUs for accelerating the compute-intensive parts of our pipeline.

3 Materials and methods

3.1 Algorithm

In the following, we explain the individual steps of the overall workflow of CARE (Fig. 1). In addition, there is an initial stage for database construction and a post-processing stage for consensus voting from different corrections of the same read.

3.1.1 Database construction

Consider a collection of reads $R = \{r_1, \dots, r_m\}$ and an input parameter k . Unless otherwise stated, we assume each read to be of equal length l . In a pre-processing step, all ambiguous nucleotides can be randomly replaced by A, C, G or T. This is required for reads to participate in error correction. Furthermore, let h_1, \dots, h_H be H hash functions that map k -mers to unique integers used for minhashing.

Each read $r_i \in R$ is transformed into its set of strand-neutral canonical k -mers (a canonical k -mer is the lexicographically smaller one of a k -mer and its reverse complement). Subsequently, each hash function h_j is applied to every canonical k -mer. Only the H smallest resulting integers for each hash function are kept representing the *signature* of r_i .

All signatures are inserted into a database. The database consists of H hash tables (key-value stores). The j th hash table is associated with the hash function h_j . In each bucket, we store the hash value as key and the IDs of all reads containing this hash value at position j of their signature. If a bucket exceeds a certain number of reads (default: $2.5 \times \text{estimated_coverage}$), it is considered uninformative and is removed from the hash table.

3.1.2 Database querying

We consider each $r_i \in R$ as anchor read. To find an initial set of candidate reads $C(r_i)$, the read signature of r_i is calculated again and used to query the database. The query returns a list of read IDs for each hash table j , i.e. these reads have the identical minhash signature at the position j . $C(r_i)$ is the union of the returned read IDs for all $j \in \{1, \dots, H\}$.

3.1.3 Filtering of candidate reads

Since minhashing is a probabilistic measure, it does not necessarily guarantee sufficiently high degrees of similarity between the anchor r_i and every read in $C(r_i)$. Thus, we further refine the set of candidates to obtain a potentially smaller set of highly similar candidates $F(r_i)$. Ideally, the filtered set $F(r_i)$ should only contain reads with high overlaps and very few mismatches to the anchor, thus maximizing the probability that they are sequenced from the same genomic

We present an MSA-based error corrector for Illumina data—called CARE—that can efficiently scale towards human-size genomes. CARE is based on a variant of *minhashing* to quickly find a set of candidate reads which are similar to a query read with high probability. This allows for fast construction of high-quality MSAs that are further refined and then inspected in detail to perform context-aware correction. We demonstrate that our approach is beneficial to

1. runtime and scalability,
2. reduction of false-positives corrections and
3. downstream analysis using *de novo* genome assembly

In particular, the speed of CARE is comparable to the fastest state-of-the-art k -mer spectrum methods but produces significantly fewer false-positive corrections while maintaining a competitive number of true positives. It efficiently scales towards large sequencing read collections (such as human genome NGS data) resulting in significantly shorter runtimes than the recent MSA-based methods while still producing around one order-of-magnitude fewer false positives on average. In particular, low numbers of false positives (i.e. fewer newly introduced errors) are an attractive feature for downstream analysis. We show that prior application of CARE results in superior *de novo* assembly results for several datasets compared to other error correction tools.

2 Approach

Consider a massive collection of sequencing reads. Our aim is to efficiently calculate similarities between them to construct meaningful MSAs. A brute-force approach that computes all pairwise alignments is computationally infeasible due to the large data sizes. MSA-based error correction approaches often consider *anchor* reads. Each anchor is aligned to a set of candidate reads to find a consensus sequence which is used for correction. A key aspect is the selection of candidate reads, which needs to be accurate and quick. For example, Coral (Salmela and Schröder, 2011) and ECHO (Kao *et al.*, 2011) identify candidate reads that share at least one exact k -mer with the anchor. Unfortunately, this simple method has low sensitivity for large values of k , since many relevant candidate reads will be missed and has low selectivity for small values of k , since the number of identified candidate reads grows enormously.

We address this problem using a variant of *minhashing* to quickly find a set of candidate reads which are very similar to the anchor read with high probability. Read *signatures* are generated by applying a number of hash functions to each k -mer of a read and choosing the smallest hash value for each function. These signatures provide a memory-reduced representation which can be used for string comparison: the relative intersection ratios of two read signatures approximate the true Jaccard Index evaluated on their whole k -mer space. CARE is thus able to use a relatively small value of k ($k = 20$ by default) to identify candidate reads with both high selectivity and high sensitivity.

Our approach requires only a standard workstation with a sufficient amount of main memory. We take advantage of commonly available parallelism by exploiting multi-threading on modern multi-core CPUs. In addition, we provide a CUDA version to exploit GPUs for accelerating the compute-intensive parts of our pipeline.

3 Materials and methods

3.1 Algorithm

In the following, we explain the individual steps of the overall workflow of CARE (Fig. 1). In addition, there is an initial stage for database construction and a post-processing stage for consensus voting from different corrections of the same read.

3.1.1 Database construction

Consider a collection of reads $R = \{r_1, \dots, r_m\}$ and an input parameter k . Unless otherwise stated, we assume each read to be of equal length l . In a pre-processing step, all ambiguous nucleotides can be randomly replaced by A, C, G or T. This is required for reads to participate in error correction. Furthermore, let b_1, \dots, b_H be H hash functions that map k -mers to unique integers used for minhashing.

Each read $r_i \in R$ is transformed into its set of strand-neutral canonical k -mers (a canonical k -mer is the lexicographically smaller one of a k -mer and its reverse complement). Subsequently, each hash function b_j is applied to every canonical k -mer. Only the H smallest resulting integers for each hash function are kept representing the *signature* of r_i .

All signatures are inserted into a database. The database consists of H hash tables (key-value stores). The j th hash table is associated with the hash function b_j . In each bucket, we store the hash value as key and the IDs of all reads containing this hash value at position j of their signature. If a bucket exceeds a certain number of reads (default: $2.5 \times \text{estimated_coverage}$), it is considered uninformative and is removed from the hash table.

3.1.2 Database querying

We consider each $r_i \in R$ as anchor read. To find an initial set of candidate reads $C(r_i)$, the read signature of r_i is calculated again and used to query the database. The query returns a list of read IDs for each hash table j , i.e. these reads have the identical minhash signature at the position j . $C(r_i)$ is the union of the returned read IDs for all $j \in \{1, \dots, H\}$.

3.1.3 Filtering of candidate reads

Since minhashing is a probabilistic measure, it does not necessarily guarantee sufficiently high degrees of similarity between the anchor r_i and every read in $C(r_i)$. Thus, we further refine the set of candidates to obtain a potentially smaller set of highly similar candidates $F(r_i)$. Ideally, the filtered set $F(r_i)$ should only contain reads with high overlaps and very few mismatches to the anchor, thus maximizing the probability that they are sequenced from the same genomic

We present an MSA-based error corrector for Illumina data—called CARE—that can efficiently scale towards human-size genomes. CARE is based on a variant of *minhashing* to quickly find a set of candidate reads which are similar to a query read with high probability. This allows for fast construction of high-quality MSAs that are further refined and then inspected in detail to perform context-aware correction. We demonstrate that our approach is beneficial to

1. runtime and scalability,
2. reduction of false-positives corrections and
3. downstream analysis using *de novo* genome assembly

In particular, the speed of CARE is comparable to the fastest state-of-the-art k -mer spectrum methods but produces significantly fewer false-positive corrections while maintaining a competitive number of true positives. It efficiently scales towards large sequencing read collections (such as human genome NGS data) resulting in significantly shorter runtimes than the recent MSA-based methods while still producing around one order-of-magnitude fewer false positives on average. In particular, low numbers of false positives (i.e. fewer newly introduced errors) are an attractive feature for downstream analysis. We show that prior application of CARE results in superior *de novo* assembly results for several datasets compared to other error correction tools.

2 Approach

Consider a massive collection of sequencing reads. Our aim is to efficiently calculate similarities between them to construct meaningful MSAs. A brute-force approach that computes all pairwise alignments is computationally infeasible due to the large data sizes. MSA-based error correction approaches often consider *anchor* reads. Each anchor is aligned to a set of candidate reads to find a consensus sequence which is used for correction. A key aspect is the selection of candidate reads, which needs to be accurate and quick. For example, Coral (Salmela and Schröder, 2011) and ECHO (Kao *et al.*, 2011) identify candidate reads that share at least one exact k -mer with the anchor. Unfortunately, this simple method has low sensitivity for large values of k , since many relevant candidate reads will be missed and has low selectivity for small values of k , since the number of identified candidate reads grows enormously.

We address this problem using a variant of *minhashing* to quickly find a set of candidate reads which are very similar to the anchor read with high probability. Read *signatures* are generated by applying a number of hash functions to each k -mer of a read and choosing the smallest hash value for each function. These signatures provide a memory-reduced representation which can be used for string comparison: the relative intersection ratios of two read signatures approximate the true Jaccard Index evaluated on their whole k -mer space. CARE is thus able to use a relatively small value of k ($k = 20$ by default) to identify candidate reads with both high selectivity and high sensitivity.

(2017). Our approach requires only a standard workstation with a sufficient amount of main memory. We take advantage of commonly available parallelism by exploiting multi-threading on modern multi-core CPUs. In addition, we provide a CUDA version to exploit GPUs for accelerating the compute-intensive parts of our pipeline.

3 Materials and methods

3.1 Algorithm

In the following, we explain the individual steps of the overall workflow of CARE (Fig. 1). In addition, there is an initial stage for database construction and a post-processing stage for consensus voting from different corrections of the same read.

3.1.1 Database construction

Consider a collection of reads $R = \{r_1, \dots, r_m\}$ and an input parameter k . Unless otherwise stated, we assume each read to be of equal length l . In a pre-processing step, all ambiguous nucleotides can be randomly replaced by A, C, G or T. This is required for reads to participate in error correction. Furthermore, let b_1, \dots, b_H be H hash functions that map k -mers to unique integers used for minhashing.

Each read $r_i \in R$ is transformed into its set of strand-neutral canonical k -mers (a canonical k -mer is the lexicographically smaller one of a k -mer and its reverse complement). Subsequently, each hash function b_j is applied to every canonical k -mer. Only the H smallest resulting integers for each hash function are kept representing the *signature* of r_i .

All signatures are inserted into a database. The database consists of H hash tables (key-value stores). The j th hash table is associated with the hash function b_j . In each bucket, we store the hash value as key and the IDs of all reads containing this hash value at position j of their signature. If a bucket exceeds a certain number of reads (default: $2.5 \times \text{estimated_coverage}$), it is considered uninformative and is removed from the hash table.

3.1.2 Database querying

We consider each $r_i \in R$ as anchor read. To find an initial set of candidate reads $C(r_i)$, the read signature of r_i is calculated again and used to query the database. The query returns a list of read IDs for each hash table j , i.e. these reads have the identical minhash signature at the position j . $C(r_i)$ is the union of the returned read IDs for all $j \in \{1, \dots, H\}$.

3.1.3 Filtering of candidate reads

Since minhashing is a probabilistic measure, it does not necessarily guarantee sufficiently high degrees of similarity between the anchor r_i and every read in $C(r_i)$. Thus, we further refine the set of candidates to obtain a potentially smaller set of highly similar candidates $F(r_i)$. Ideally, the filtered set $F(r_i)$ should only contain reads with high overlaps and very few mismatches to the anchor, thus maximizing the probability that they are sequenced from the same genomic

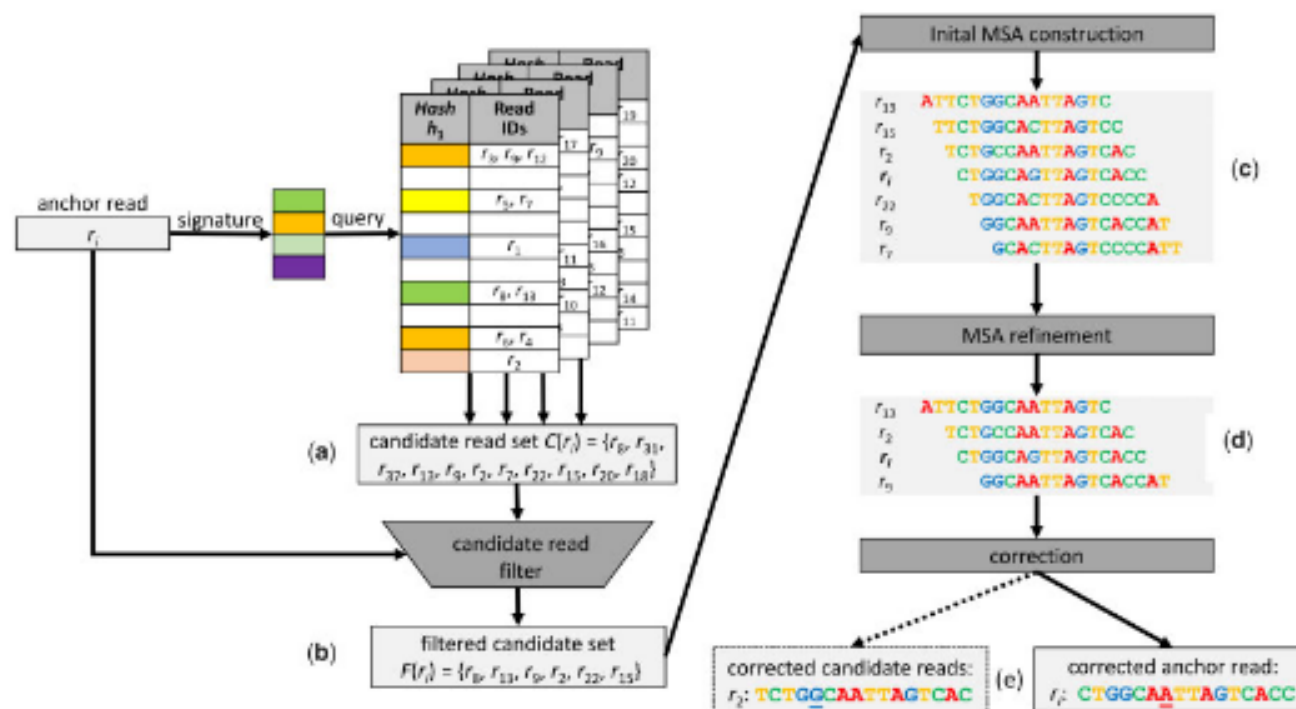


Fig. 1. Workflow of CARE: (a) The signature of an anchor read (r_i) is determined by minhashing and used to query the pre-computed hash tables. The retrieved reads form the candidate read set $C(r_i)$. (b) All reads in $C(r_i)$ are aligned to r_i . Reads with a relatively low semi-global pairwise alignment quality are removed, resulting in the filtered set of candidate reads ($F(r_i)$). (c) The initial MSA is constructed around the center r_i using $F(r_i)$. (d) The MSA is refined by removing candidate reads with a significantly different pattern from the anchor (i.e. r_{15} , r_{22} , r_7 in the example). (e) The anchor read (the seventh nucleotide in r_i in the example) and optionally some of the candidates are corrected (the fifth nucleotide in r_2 in the example)

region. However, the number of identified candidate reads can vary due to non-uniform coverage, repeats and high error rates. Thus, a flexible algorithmic procedure is required.

We compute a pairwise semi-global alignment between r_i and each $c \in C(r_i)$, as well as between r_i and the reverse complement of c . Dissimilar candidate reads produce low-quality alignments and are removed from $C(r_i)$. The remaining reads with high similarity are partitioned according to their alignment quality.

CARE is designed for Illumina data where substitutions are the dominant error source. Thus, we only consider gap-free semi-global pairwise alignments. These can be computed efficiently via shifted hamming distance (Xin *et al.*, 2015). We determine an optimal shift value s , $-l + \delta \leq s \leq l - \delta$, such that if c is shifted to the right by s positions relative to r_i , the overlap between r_i and c has minimal hamming distance (ϵ). Negative values of s indicate left shifts. Candidates with a shifted hamming distance greater than a threshold (default: $\epsilon > 0.2 \cdot \text{overlap}_{\text{size}}$) are removed. Note that we only consider alignments with sufficient overlap (default: $\delta = 0.3 \cdot l$).

3.1.4 Initial MSA construction

An initial MSA M is constructed using $F(r_i)$ and r_i as the center sequence in a manner similar to the well-known center star alignment approximation algorithm (Gusfield, 1997). Let $rbegin$ and $rend$ denote the column indices in M such that $r_i[0]$ is located in column $rbegin$ and $r_i[l-1]$ in column $rend$. The position of candidate reads within M is determined by their previously calculated optimal shift value s , e.g. the leftmost column occupied by a candidate is $rbegin + s$.

We store weights of A, C, G and T for each MSA column. The weight of an individual nucleotide ranges between 0.0 and 1.0 and is determined by a combination of the alignment quality of the corresponding candidate and the quality score (if the input is a FASTQ file). Thus, nucleotides with high-quality scores coming from candidate reads that are very similar to the anchor are assigned higher weights.

The consensus of a column in M is determined as the nucleotide with highest accumulative weight within the respective column. The

Table 1. Simulated (S1–S7) and real (R1–R3) datasets used for evaluation

Name	Organism	Cov.	HiSeq 2000		MiSeq v3	
			No. of reads	No. of errors	No. of reads	No. of errors
S1	<i>C.elegans</i>	30×	30.1M	22.6M	12M	48.9M
S2	<i>C.elegans</i>	60×	60.2M	45.2M	24.1M	97.9M
S3	<i>A.thaliana</i>	30×	35.8M	26.9M	14.3M	58.3M
S4	<i>A.thaliana</i>	60×	71.7M	53.8M	28.6M	116.6M
S5	Hum. Chr. 14	30×	26.5M	19.9M	10.6M	43.1M
S6	Hum. Chr. 14	60×	53.0M	39.8M	21.2M	86.2M
S7	Human	30×	914.7M	687.1M	365.9M	1488M
R1	<i>C.elegans</i>	58×	57.7M	Unknown		
R2	<i>D.melanogaster</i>	64×	75.9M	Unknown		
R3	<i>Aiptasia</i>	20×	53.5M	Unknown		

Note: Simulated HiSeq 2000 datasets have a read length of 100 and an error rate of 0.75%. Simulated MiSeq v3 datasets have a read length of 250 with an error rate of 1.62%. No. of errors indicates the total number of erroneous nucleotides. The real datasets have a read length of 101 and are downloaded from the NCBI SRA using the accession numbers SRR543736, SRR988075, SRR606428.

Figure 2 shows the average relative improvement of CARE over all datasets compared to each other tool for each category. Absolute numbers of TPs and FPs for each tool, each simulated dataset and each sequencing technology (HiSeq2000 and MiSeq v3) are available in Supplementary File, Supplementary Section S5, Supplementary Tables S4 and S5. CARE achieves the lowest number of FPs and the lowest FP-rate for all tested datasets while maintaining a competitive number of TPs. On average for the HiSeq2000 datasets the number of FPs and the FP-rate of CARE is over one order-of-magnitude smaller compared to Muskett, SGA, Bcool and Lighter and close to one order-of-magnitude smaller compared to Karect and BFC. On average for the MiSeq v3 datasets, the number of FPs and the FP-rate of CARE is two orders-of-magnitude smaller compared to Muskett, over one order-of-magnitude smaller SGA, Karect and Lighter and over five times smaller compared to BFC. On average the number of TPs of CARE is highly competitive, falling short by only approximately 4% compared to the best TP values for the HiSeq datasets (compared to SGA, Karect, BFC) and falling less than 1% short for the MiSeq datasets (compared to Karect). To analyze the influence of selected parameter values, we present

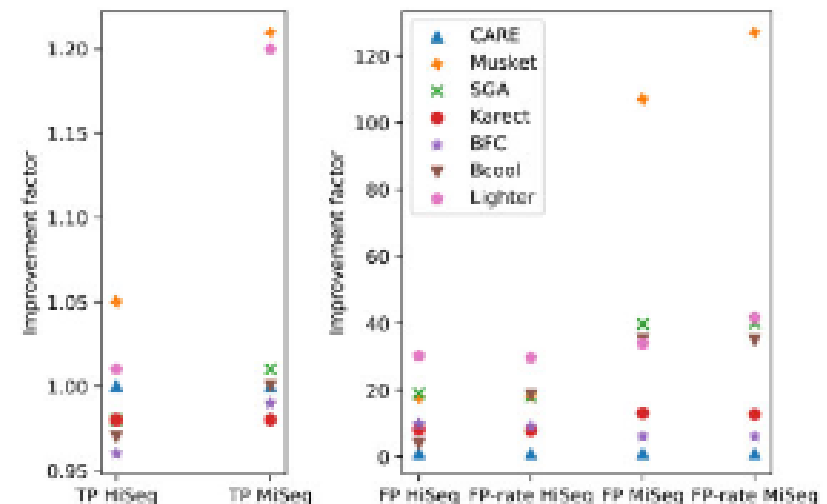


Fig. 2. Average improvement factor of CARE over selected tools for the number of true positive corrections (TP), the number of false-positive corrections (FP) and the number of false-positive corrections per one million corrections (FP-rate). A value greater than one indicates advantage of CARE.

Protein isoelectric point calculator

Home	Theory	Datasets	Algorithms	Results	Links
------	--------	----------	------------	---------	-------

Datasets used for isoelectric point optimization

Dataset	Initial no. entries	No. entries with sequence and pI	No. entries after removing outliers	No. entries after removing duplicates
Gauci et al.	5,758	5,758	NA	NA
PHENYX	7,582	7,582	NA	NA
SEQUEST	7,629	7,629	NA	NA
IPC_peptide	-	20,969	20,969	16,882 [25] [75]
SWISS-2DPAGE	2,530	1,054	1,029	982
PIP-DB	4,947	2,427	2,254	1,307
IPC_protein	-	3,481	3,283	2,324 [25] [75]

Note:

- all datasets presented in the table are available as hyperlinks, final datasets were divided randomly into 75% training and 25% testing subsets (hyperlinks denoted

Scientific article structure

Always provide

Raw data

Source code

Data

There is no excuse to write in the article things like that:

*The data that support the findings of this study are available from the corresponding author, [author initials], upon **reasonable** request*


Data not shown

etc.

Data

nature immunology

View all journals

Search 

Log in

Explore content ▾

About the journal ▾

Publish with us ▾


Subscribe

Sign up for alerts 

RSS feed

[nature](#) > [nature immunology](#) > [articles](#) > articleArticle | [Published: 20 April 2023](#)

TREM2 macrophages drive NK cell paucity and dysfunction in lung cancer

[Matthew D. Park](#), [Ivan Reyes-Torres](#), [Jessica LeBerichel](#), [Pauline Hamon](#), [Nelson M. LaMarche](#), [Samarth Hegde](#), [Meriem Belabed](#), [Leanna Troncoso](#), [John A. Grout](#), [Assaf Magen](#), [Etienne Humblin](#), [Achuth Nair](#), [Martina Molgora](#), [Jinchao Hou](#), [Jenna H. Newman](#), [Adam M. Farkas](#), [Andrew M. Leader](#), [Travis Dawson](#), [Darwin D'Souza](#), [Steven Hamel](#), [Alfonso Rodriguez Sanchez-Paulete](#), [Barbara Maier](#), [Nina Bhardwaj](#), [Jerome C. Martin](#), ... [Miriam Merad](#)  [+ Show authors](#)

[Nature Immunology](#) (2023) | [Cite this article](#)

Access to this article via **ICM Warsaw University** is not available.

[Change institution](#)[Buy or subscribe](#)

Sections

Figures

References

[Abstract](#)

Code availability

All murine sequencing code will be made publicly available ([GSE184304](#), [GSE184309](#) and [GSE184317](#)). No custom code was generated for this study. Code for generating figures can be provided upon reasonable request.


Data

nature immunology

IF 31.25

[View all journals](#)[Search](#)[Log in](#)[Explore content](#)[About the journal](#)[Publish with us](#)[Subscribe](#)[Sign up for alerts](#)[RSS feed](#)[nature](#) > [nature immunology](#) > [articles](#) > articleArticle | [Published: 20 April 2023](#)

TREM2 macrophages drive NK cell paucity and dysfunction in lung cancer

[Matthew D. Park](#), [Ivan Reyes-Torres](#), [Jessica LeBerichel](#), [Pauline Hamon](#), [Nelson M. LaMarche](#), [Samarth Hegde](#), [Meriem Belabed](#), [Leanna Troncoso](#), [John A. Grout](#), [Assaf Magen](#), [Etienne Humblin](#), [Achuth Nair](#), [Martina Molgora](#), [Jinchao Hou](#), [Jenna H. Newman](#), [Adam M. Farkas](#), [Andrew M. Leader](#), [Travis Dawson](#), [Darwin D'Souza](#), [Steven Hamel](#), [Alfonso Rodriguez Sanchez-Paulete](#), [Barbara Maier](#), [Nina Bhardwaj](#), [Jerome C. Martin](#), ... [Miriam Merad](#)  [+ Show authors](#)

[Nature Immunology](#) (2023) | [Cite this article](#)

Access to this article via **ICM Warsaw University** is not available.

[Change institution](#)[Buy or subscribe](#)[Sections](#)[Figures](#)[References](#)[Abstract](#)

Code availability

All murine sequencing code will be made publicly available ([GSE184304](#), [GSE184309](#) and [GSE184317](#)). No custom code was generated for this study. Code for generating figures can be provided upon reasonable request.

[nature](#) > [articles](#) > [article](#)

Article | [Published: 05 April 2023](#)

Large-scale mapping and mutagenesis of human transcriptional effector domains

[Nicole DelRosso](#), [Josh Tycko](#), [Peter Suzuki](#), [Cecelia Andrews](#), [Aradhana](#), [Adi Mukund](#), [Ivan Liongson](#), [Connor Ludwig](#), [Kaitlyn Spees](#), [Polly Fordyce](#), [Michael C. Bassik](#) & [Lacramioara Bintu](#) 

Nature **616**, 365–372 (2023) | [Cite this article](#)

12k Accesses | **135** Altmetric | [Metrics](#)

Code availability



The HT-recruit Analyze software for processing high-throughput recruitment assay and high-throughput protein expression assays are available on GitHub (<https://github.com/bintulab/HT-recruit-Analyze>). All custom codes used for data processing and computational analyses are available from the authors upon request.

Taking the pain out of data sharing

Despite agreeing to make raw data available, some authors fail to comply. The right strategies and platforms can ease the task.

Original Article

Many researchers were not compliant with their published data sharing statement: a mixed-methods study

[Mirko Gabelica](#)^a, [Ružica Bojčić](#)^b, [Livia Puljak](#)^c  

Results

Of 3556 analyzed articles, 3416 contained the DAS. The most frequent DAS category (42%) indicated that the data sets are available on reasonable request. Among 1792 manuscripts in which the DAS indicated that authors are willing to share their data, 1669 (93%) authors either did not respond or declined to share their data with us. Among 254 (14%) of 1792 authors who responded to our query for data sharing, only 123 (6.8%) provided the requested data.

<https://doi.org/10.1038/d41586-022-03133-5>

<https://doi.org/10.1016/j.jclinepi.2022.05.019>

Data

There is no excuse to write in the article things like that:

*The data that support the findings of this study are available from the corresponding author, [author initials], upon **reasonable** request*

Data not shown

etc.

and you too as a researcher do not want to get an email regarding the data from the project that finished 10 years ago.

Data formats



Data & Result repositories

Data & Result repositories

**We did some project
&
finally, we have some results**

Data & Result repositories

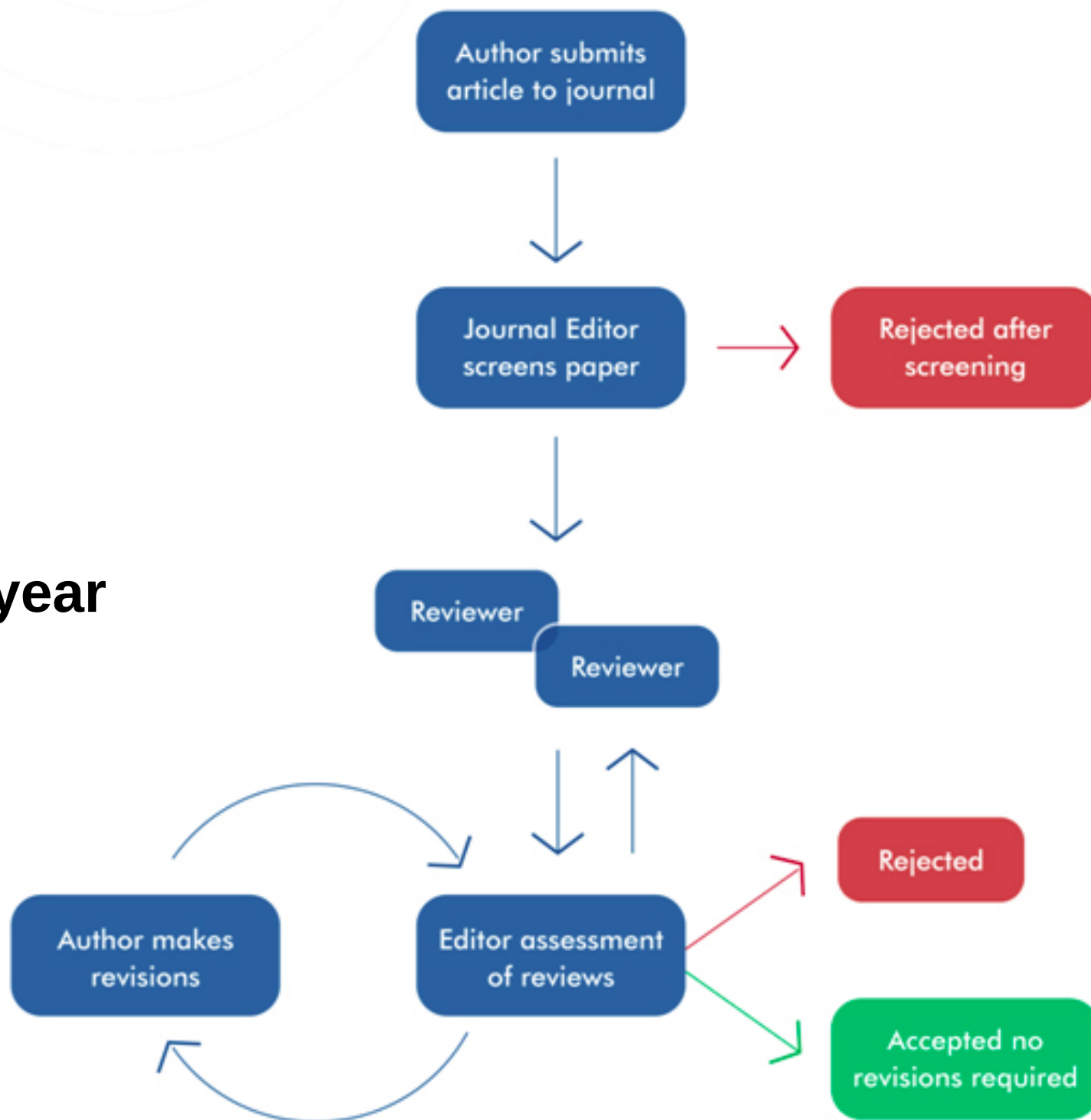


Data & Result repositories





0.5-1 year



Data repositories

CARE: context-aware sequencing read error correction

Felix Kallenborn  *, Andreas Hildebrandt and Bertil Schmidt*

Department of Computer Science, Johannes Gutenberg University, Mainz 55122, Germany

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on May 8, 2020; revised on July 14, 2020; editorial decision on August 6, 2020; accepted on August 14, 2020

Abstract

Motivation: Error correction is a fundamental pre-processing step in many Next-Generation Sequencing (NGS) pipelines, in particular for *de novo* genome assembly. However, existing error correction methods either suffer from high false-positive rates since they break reads into independent *k*-mers or do not scale efficiently to large amounts of sequencing reads and complex genomes.

Results: We present CARE—an alignment-based scalable error correction algorithm for Illumina data using the concept of minhashing. Minhashing allows for efficient similarity search within large sequencing read collections which enables fast computation of high-quality multiple alignments. Sequencing errors are corrected by detailed inspection of the corresponding alignments. Our performance evaluation shows that CARE generates significantly fewer false-positive corrections than state-of-the-art tools (Musket, SGA, BFC, Lighter, Bcool, Karect) while maintaining a competitive number of true positives. When used prior to assembly it can achieve superior *de novo* assembly results for a number of real datasets. CARE is also the first multiple sequence alignment-based error corrector that is able to process a human genome Illumina NGS dataset in only 4 h on a single workstation using GPU acceleration.

Availability and implementation: CARE is open-source software written in C++ (CPU version) and in CUDA/C++ (GPU version). It is licensed under GPLv3 and can be downloaded at <https://github.com/fkallen/CARE>.

Contact: kallenborn@uni-mainz.de or bertil.schmidt@uni-mainz.de,

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

Data repositories

CARE: context-aware sequencing read error correction

Felix Kallenborn  *, Andreas Hildebrandt and Bertil Schmidt*

Department of Computer Science, Johannes Gutenberg University, Mainz 55122, Germany

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on May 8, 2020; revised on July 14, 2020; editorial decision on August 6, 2020; accepted on August 14, 2020

Abstract

Motivation: Error correction is a fundamental pre-processing step in many Next-Generation Sequencing (NGS) pipelines, in particular for *de novo* genome assembly. However, existing error correction methods either suffer from high false-positive rates since they break reads into independent *k*-mers or do not scale efficiently to large amounts of sequencing reads and complex genomes.

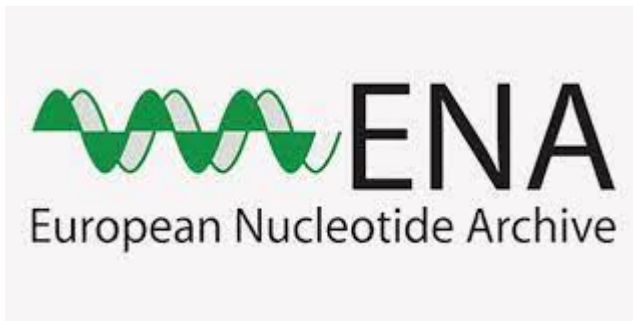
Results: We present CARE—an alignment-based scalable error correction algorithm for Illumina data using the concept of minhashing. Minhashing allows for efficient similarity search within large sequencing read collections which enables fast computation of high-quality multiple alignments. Sequencing errors are corrected by detailed inspection of the corresponding alignments. Our performance evaluation shows that CARE generates significantly fewer false-positive corrections than state-of-the-art tools (Musket, SGA, BFC, Lighter, Bcool, Karect) while maintaining a competitive number of true positives. When used prior to assembly it can achieve superior *de novo* assembly results for a number of real datasets. CARE is also the first multiple sequence alignment-based error corrector that is able to process a human genome Illumina NGS dataset in only 4 h on a single workstation using GPU acceleration.

Availability and implementation: CARE is open-source software written in C++ (CPU version) and in CUDA/C++ (GPU version). It is licensed under GPLv3 and can be downloaded at <https://github.com/fkallen/CARE>.

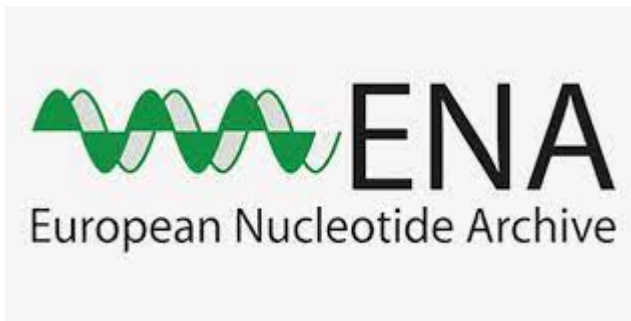
Contact: kallenborn@uni-mainz.de or bertil.schmidt@uni-mainz.de,

Supplementary information [Supplementary data](#) are available at *Bioinformatics* online.

Data repositories



Data repositories



RepOD
Repository for Open Data



<https://www.nature.com/sdata/policies/repositories>



Article

Molecular Characterization of a DNA Polymerase from *Thermus thermophilus* MAT72 Phage vB_Tt72: A Novel Type-A Family Enzyme with Strong Proofreading Activity

Sebastian Dorawa ¹, Olesia Werbowy ¹, Magdalena Plotka ¹, Anna-Karina Kaczorowska ²,
Joanna Makowska ³, Lukasz P. Kozlowski ⁴, Olafur H. Fridjonsson ⁵, Gudmundur O. Hreggvidsson ^{5,6},
Arnthór Aevarsson ⁵ and Tadeusz Kaczorowski ^{1,*}

YELLOW

3'-5' exonuclease (1-201)

BLUE

Palm (202-229, 337-365,
376-386, 606-703)

GREEN&BLUE

DNA

GREEN

Thumb (230-336,
366-375)

MAGENTA

Fingers (387-605)





RepOD

Repository for Open Data

Search

About repository

Support

English

Sign Up

Log In

Repository Project (2)

Department (1)

Laboratory (1)

Publication Date

2023 (177)

2022 (155)

2021 (92)

2020 (85)

2017 (27)

More...

Author Name

Kamiński, Radosław (15)

Kozłowski, Łukasz P. (12)

Kozłowski, Łukasz (11)

Dominiak, Paulina Maria (9)

Jastrzębska, Aneta (9)

More...

Text: 80 == Font: A A A Contrast: Aa Aa



Light-sheet fluorescence microscopy images of optically cleared and c-Fos immunolabeled mouse brain

Apr 21, 2023

Legutko, Diana; Stefaniuk, Marzena; Pawłowska, Monika; Bednarek, Sylwia; Majka, Piotr; Wójcik, Daniel K; Kaczmarek, Leszek, 2023, "Light-sheet fluorescence microscopy images of optically cleared and c-Fos immunolabeled mouse brain", <https://doi.org/10.18150/NIDUBW>, RepOD, V1



University of Gdańsk (Uniwersytet Gdański)

Apr 21, 2023



Synthetic social structure of Poland

Apr 20, 2023 - Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw

Epidemiological Model Team, 2023, "Synthetic social structure of Poland", <https://doi.org/10.18150/OZIECO>, RepOD, V1



Kazimierz Wielki University in Bydgoszcz (Uniwersytet Kazimierza Wielkiego)

Apr 20, 2023

Kolekcja Repozytorium Danych Badawczych Uniwersytetu Kazimierza Wielkiego umożliwia gromadzenie,



Supplementary Data for "Molecular characterization of a DNA polymerase from *Thermus thermophilus* MAT72 phage vB_Tt72, a novel type-A family enzyme with strong proofreading activity" by Dorawa S., Werbowy O., Plotka M., Kaczorowska A.-K., Makowska J., Kozłowski L.P., Fridjonsson O.H., Hreggvidsson G.O., Aevansson A., Kaczorowski T.

Version 1.0

Kozłowski, Lukasz P., 2022, "Supplementary Data for "Molecular characterization of a DNA polymerase from *Thermus thermophilus* MAT72 phage vB_Tt72, a novel type-A family enzyme with strong proofreading activity" by Dorawa S., Werbowy O., Plotka M., Kaczorowska A.-K., Makowska J., Kozłowski L.P., Fridjonsson O.H., Hreggvidsson G.O., Aevansson A., Kaczorowski T.", <https://doi.org/10.18150/0D4BNT>, RepOD, V1

Cite Dataset

[Learn about Data Citation Standards.](#)

Description

Bioinformatics results related to the article "Molecular characterization of a DNA polymerase from *Thermus thermophilus* MAT72 phage vB_Tt72, a novel type-A family enzyme with strong proofreading activity"

```
.
├── crude_refinement_models          #models from GalaxyRefine
│   ├── model_1.pdb
│   ├── model_2.pdb
│   ├── model_3.pdb
│   ├── model_4.pdb
│   └── model_5.pdb
├── DNA_polymerase_phage_Tt72_model_after_refinement.pdb #final model - PDB format
├── DNA_polymerase_phage_Tt72_model_after_refinement.png #final model - USCF Chimera v.1.15 session
├── DNA_polymerase_phage_Tt72_model_after_refinement.py
├── DNA_polymerase_phage_Tt72_model_after_refinement_section1.png
├── DNA_polymerase_phage_Tt72_model_after_refinement_section2.png
├── DNA_polymerase_phage_Tt72_model_before_refinement.pdb
├── DNA_polymerase_phage_Tt72_model_before_refinement.py
├── Fig2A.odg
├── Fig2A.png
├── README.txt
├── Tt72_alignment.fas              #alignment
├── Tt72.avi                       #Tt72 model animation
└── Tt72.fasta                     #protein sequence
```

(2022)

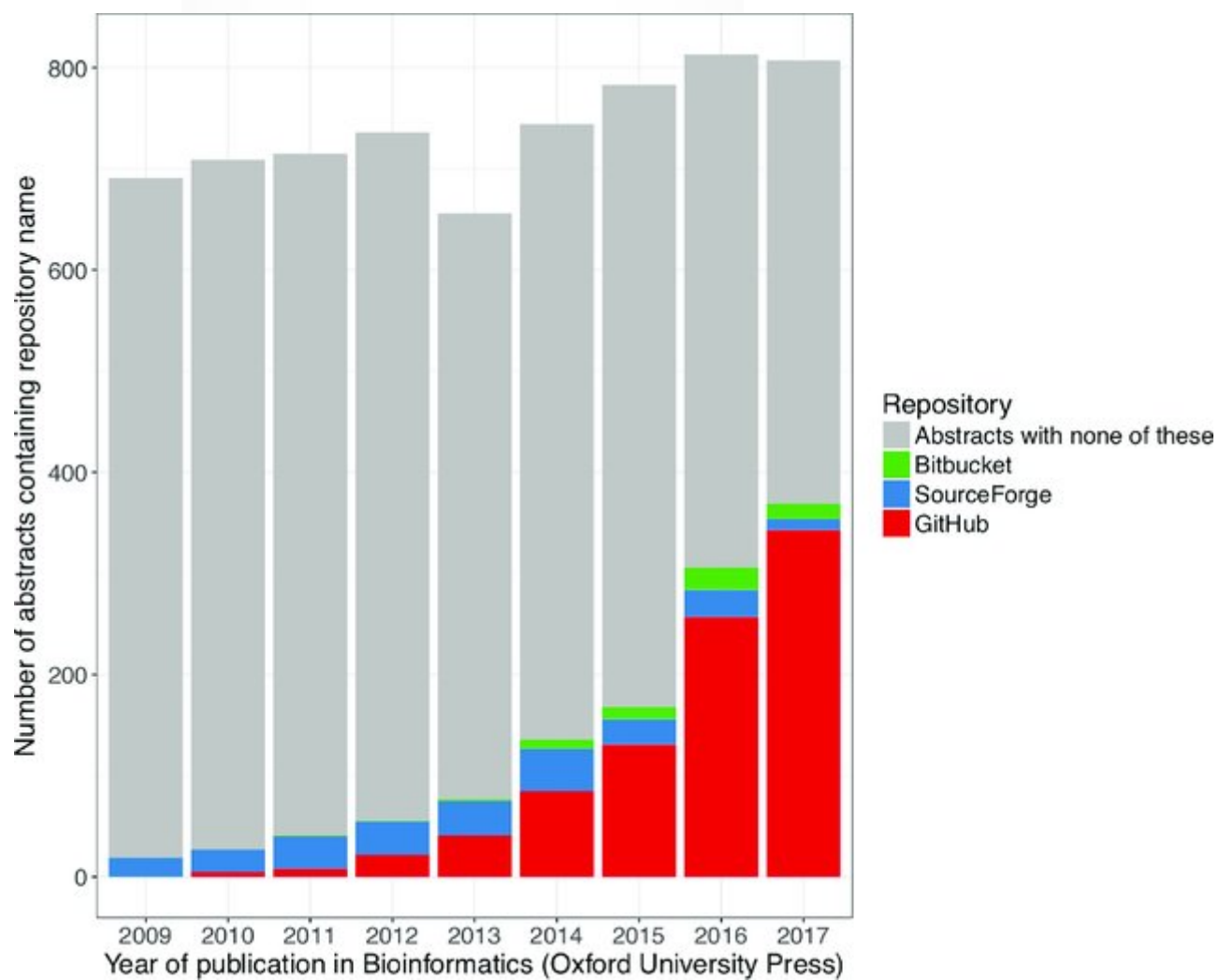
Subject

Computer and Information Science; Medicine, Health and Life Sciences

Code repositories



Code repositories



Code



Code

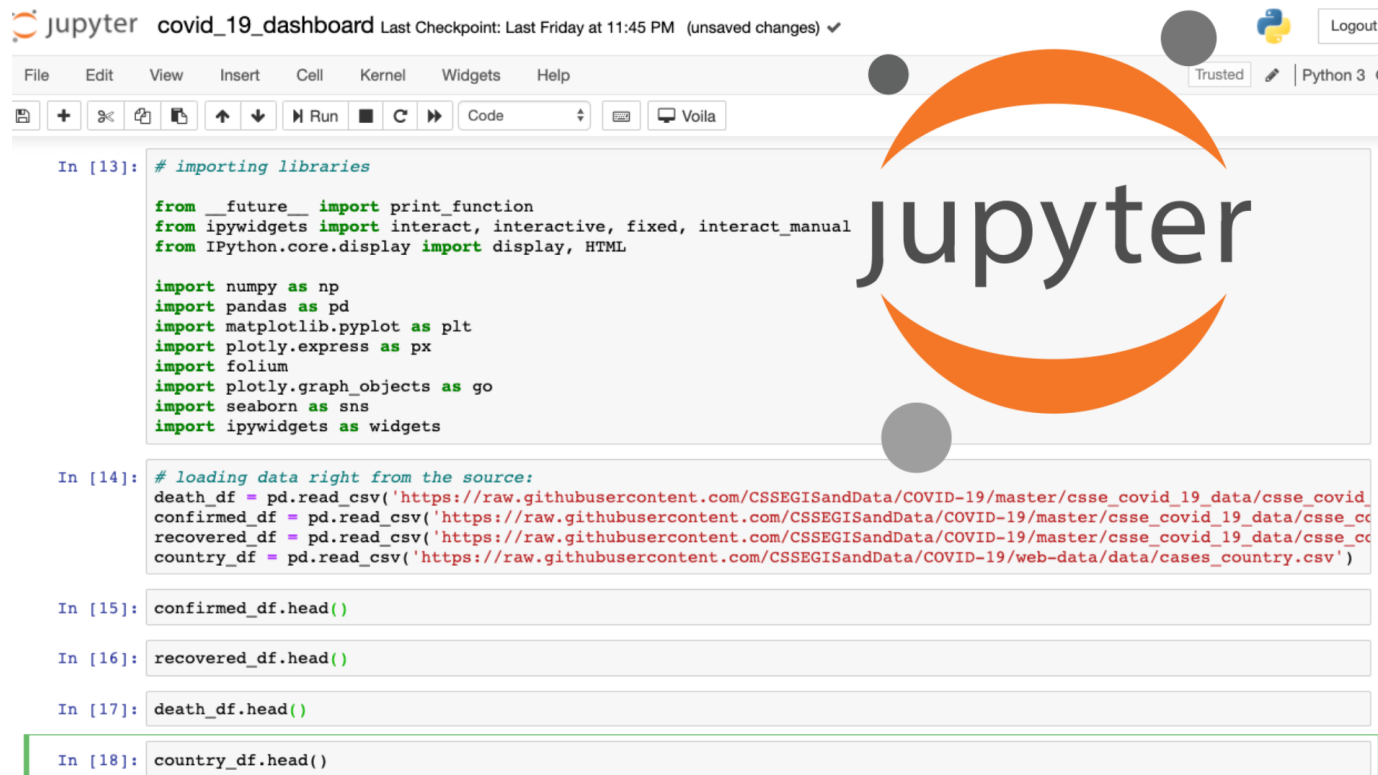


Code & Analysis

Source code of the program

but also

Scripts generating tables and figures



The image shows a JupyterLab interface with a notebook titled 'covid_19_dashboard'. The notebook contains several code cells. The first cell imports various libraries including numpy, pandas, matplotlib, plotly, folium, and seaborn. The second cell loads data from GitHub repositories into four DataFrames: death_df, confirmed_df, recovered_df, and country_df. The subsequent three cells display the first few rows of each DataFrame using the .head() method.

```
In [13]: # importing libraries

from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
from IPython.core.display import display, HTML

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import folium
import plotly.graph_objects as go
import seaborn as sns
import ipywidgets as widgets

In [14]: # loading data right from the source:
death_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_data/country_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv')

In [15]: confirmed_df.head()

In [16]: recovered_df.head()

In [17]: death_df.head()

In [18]: country_df.head()
```

Code & Analysis

Source code of the program

but also

Scripts generating tables and figures



Code & Analysis

Source code of the program

but also

Scripts generating tables and figures



Why do reproducible research?

- to show evidence of the correctness of your results**
- to enable others to make use of our methods and results**

What are the principles of reproducible research?

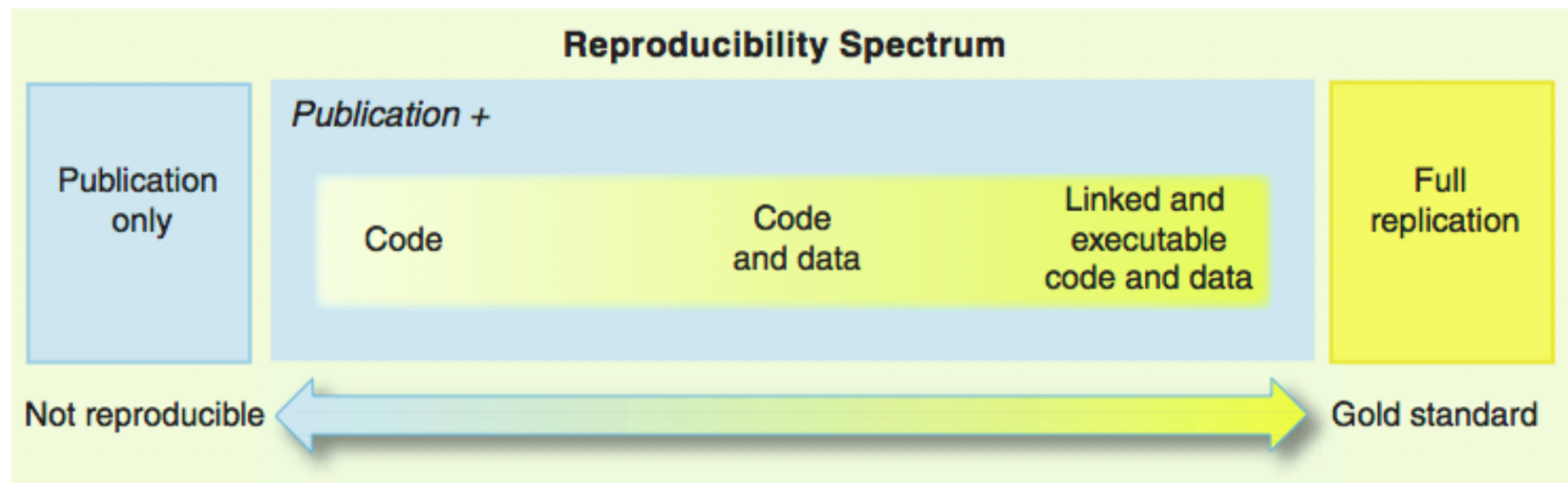
An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.

(Claerbout and Karrenbach 1992)

What are the principles of reproducible research?

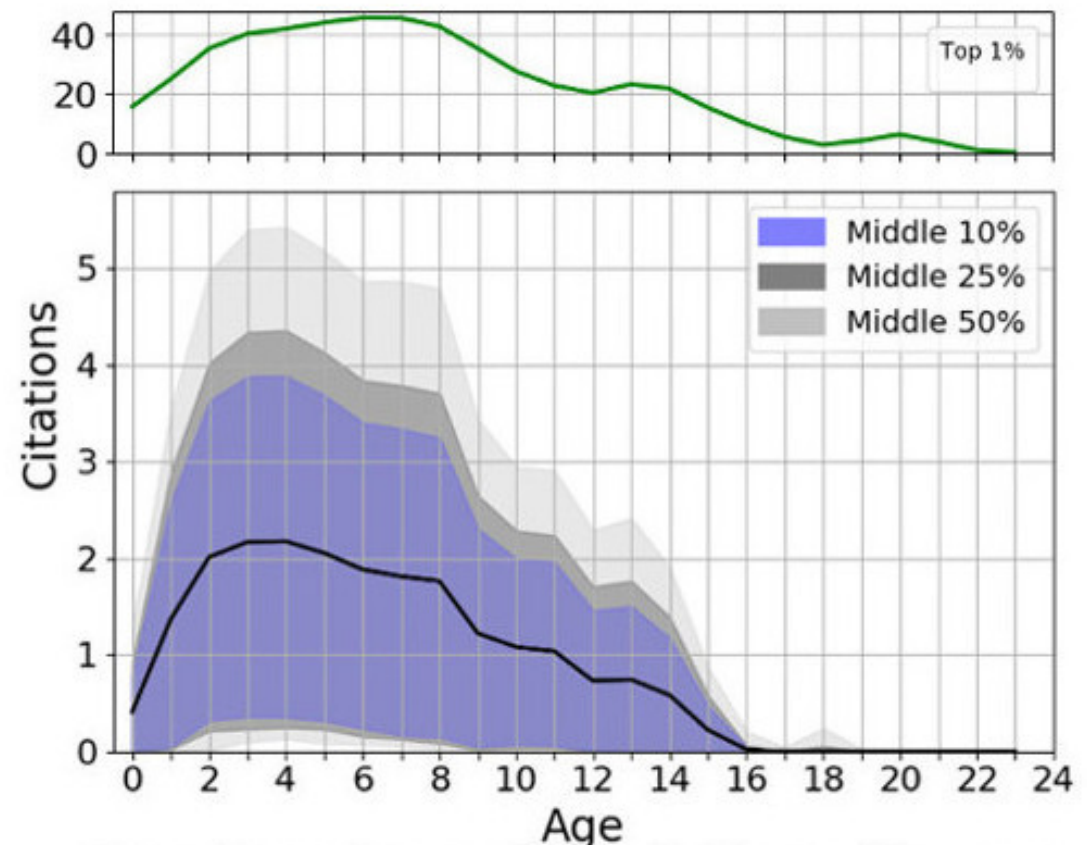
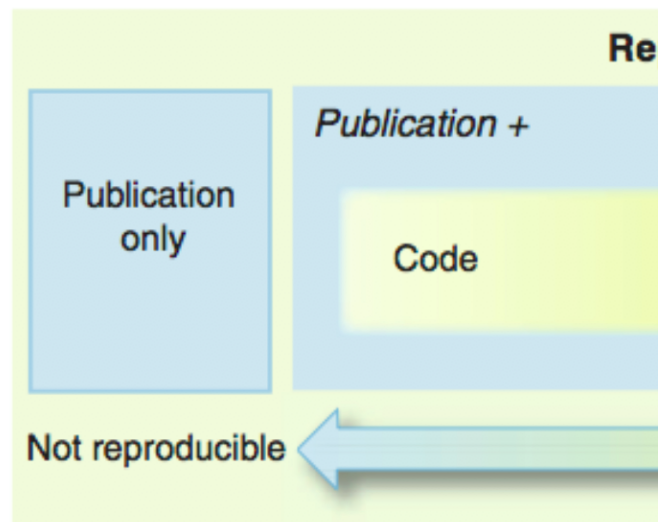
An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.

(Claerbout and Karrenbach 1992)



What are the principles of reproducible research?

The main outcome of scientific projects remain the publication, but its impact is limited up to 10-15 years

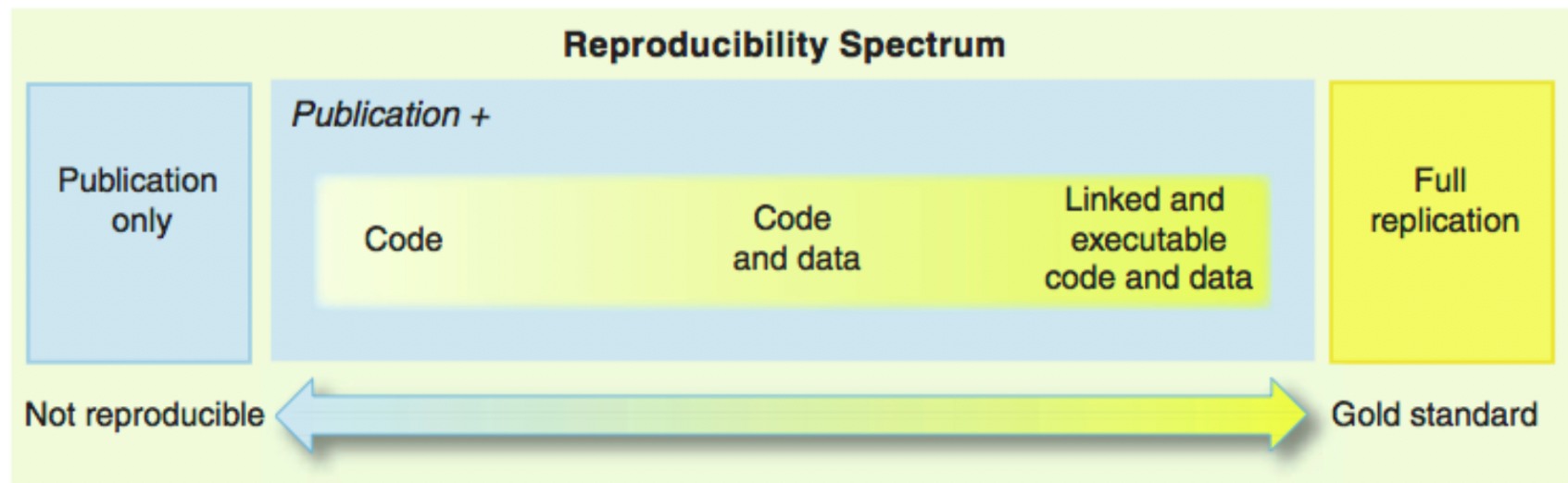


Time dependency of the citations with respect to the age of the article (most of the citations are received within a 2-5 year period)

What are the principles of reproducible research?

An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.

(Claerbout and Karrenbach 1992)



What are the different kinds of reproducible research?

- **Computational reproducibility:** when detailed information is provided about code, software, hardware and implementation details.
- **Empirical reproducibility:** when detailed information is provided about non-computational empirical scientific experiments and observations. In practise this is enabled by making data freely available, as well as details of how the data was collected.
- **Statistical reproducibility:** when detailed information is provided about the choice of statistical tests, model parameters, threshold values, etc. This mostly relates to pre-registration of study design to prevent p-value hacking and other manipulations.

What needs to be reproducible?

- **Actual results themselves, which includes:**
 - **Tables**
 - **Visualizations/figures/graphs**
 - **Values reported in the text**
- **The statistical evidence in support of the findings (e.g., p-values, confidence intervals, credible intervals)**

- **Release the code**
- **Students/Developers/You will leave, plan for it**
- **Create permanent email addresses**
- **Create project websites**
- **Use a source code control system**
- **Backup your code**
- **Resolve licensing issues**
- **Plan for longevity**
- **Avoid cool but unusual designs/techniques/programming languages, etc.**

Step 1: Before data analysis

- ☐ Are raw data safely stored in multiple locations using multiple media?
- ☐ Are final data stored in a portable, non-proprietary format?
- ☐ Are final data formatted appropriately for analysis?
- ☐ Are data paired with adequate metadata?

Step 1: Before data analysis

- ☐ Are raw data safely stored in multiple locations using multiple media?
- ☐ Are final data stored in a portable, non-proprietary format?
- ☐ Are final data formatted appropriately for analysis?
- ☐ Are data paired with adequate metadata?



Step 2: During data analysis

- ☐ Is code clean, readable, and appropriately formatted?
- ☐ Is code thoroughly commented?
- ☐ Have data and code been reviewed by at least one collaborator or friend?
- ☐ Have all software versions and computing environments been documented?

Step 1: Before data analysis

- ☐ Are raw data safely stored in multiple locations using multiple media?
- ☐ Are final data stored in a portable, non-proprietary format?
- ☐ Are final data formatted appropriately for analysis?
- ☐ Are data paired with adequate metadata?



Step 2: During data analysis

- ☐ Is code clean, readable, and appropriately formatted?
- ☐ Is code thoroughly commented?
- ☐ Have data and code been reviewed by at least one collaborator or friend?
- ☐ Have all software versions and computing environments been documented?



Step 3: After data analysis

- ☐ Are explicit instructions on locating data, metadata, and code detailed in the manuscript?
- ☐ Will data, metadata, and code be shared together at a permanent site?

Thank you for your time
and
See you at the next lecture

Any other
questions & comments

lukaskoz@mimuw.edu.pl