UNIVERSITÀ DI PISA

Metagenomic analysis through the eBWT

Veronica Guerrini Giovanna Rosone

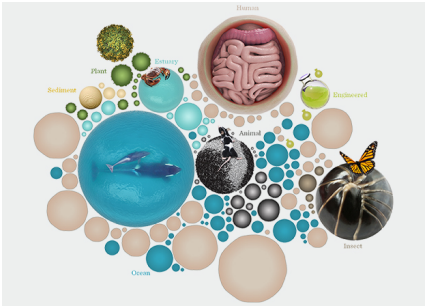
University of Pisa, Italy

Supported by the Project MIUR-SIR CMACBioSeq
"Combinatorial Methods for Analysis and Compression of Biological Sequences"

BITS 2019
Bioinformatics Italian Society Annual Meeting
Palermo, June 26 - 28, 2019

Introduction

Metagenomics is the study of genetic material collected from the environment



[Illustration: Spencer Phillips, EMBL-EBI]

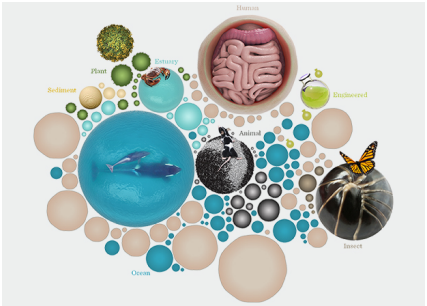
Aim to explore the relations between the microbes and their habitats

Applications. Clinical microbiology, plant-microbe interactions, monitoring pollution, sustainability, ecology, ...

Goal: Identify the taxon of each short read

Introduction

Metagenomics is the study of genetic material collected from the environment



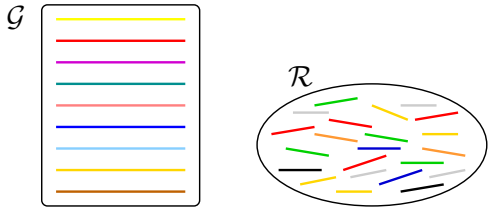
[Illustration: Spencer Phillips, EMBL-EBI]

Aim to explore the relations between the microbes and their habitats

Applications. Clinical microbiology, plant-microbe interactions, monitoring pollution, sustainability, ecology, ...

Goal: Identify the taxon of each short read

Our approach based on (e)BWT



We introduce an alignment-free and assembly-free strategy...

... by using the properties of

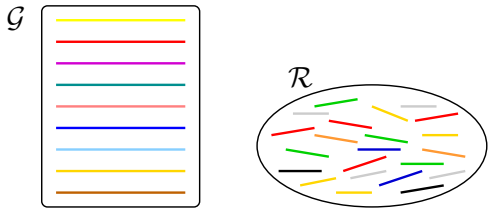
- the Burrows-Wheeler Transform (BWT) of a string,
- an extension of BWT to a multiset of strings (eBWT).

Definition

Given a string v (resp. a string collection S), the Burrows-Wheeler Transform (resp. extended BWT)[a] is a reversible transformation that produces a permutation of the symbols of v (resp. S), defined over an ordered alphabet.

[a] Burrows and D. Wheeler, A block sorting lossless data compression algorithm, Technical Report 124, Digital Equipment Corporation, 1994
S. Mantaci, A. Restivo, G. R., M. Sciortino: An extension of the Burrows-Wheeler Transform. TCS 2007
M.Bauer, A. Cox, G. R.: Lightweight algorithms for constructing and inverting the BWT of string collections. TCS 2013

Our approach based on (e)BWT



We introduce an alignment-free and assembly-free strategy...
... by using the properties of

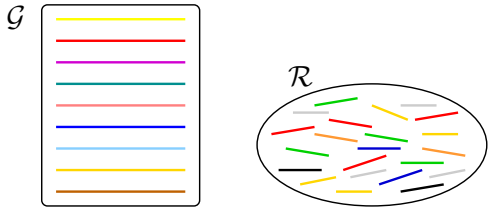
- the Burrows-Wheeler Transform (BWT) of a string,
- an extension of BWT to a multiset of strings (eBWT).

Definition

Given a string v (resp. a string collection S), the Burrows-Wheeler Transform (resp. extended BWT)[a] is a reversible transformation that produces a permutation of the symbols of v (resp. S), defined over an ordered alphabet.

[a] Burrows and D. Wheeler, A block sorting lossless data compression algorithm, Technical Report 124, Digital Equipment Corporation, 1994
S. Mantaci, A. Restivo, G. R., M. Sciortino: An extension of the Burrows-Wheeler Transform. TCS 2007
M.Bauer, A. Cox, G. R.: Lightweight algorithms for constructing and inverting the BWT of string collections. TCS 2013

Our approach based on (e)BWT



We introduce an alignment-free and assembly-free strategy...
... by using the properties of

- the Burrows-Wheeler Transform (BWT) of a string,
- an extension of BWT to a multiset of strings (eBWT).

Definition

Given a string v (resp. a string collection S), the Burrows-Wheeler Transform (resp. extended BWT)[a] is a reversible transformation that produces a permutation of the symbols of v (resp. S), defined over an ordered alphabet.

[a] Burrows and D. Wheeler, A block sorting lossless data compression algorithm, Technical Report 124, Digital Equipment Corporation, 1994
S. Mantaci, A. Restivo, G. R., M. Sciortino: An extension of the Burrows-Wheeler Transform. TCS 2007
M.Bauer, A. Cox, G. R.: Lightweight algorithms for constructing and inverting the BWT of string collections. TCS 2013

How does EBWT work?

Let S $= \{S_1, S_2, \ldots, S_m\}$ be a collection of strings.

- Append an end-marker \$, where $\$_i < A < C < G < T$ (we add subscripts $\$_i$ only for illustrative purposes), to each string in S by obtaining a new collection S′.

String collection S

	0	1	2	3	4	5
S_1	G	C	C	A	A	C
S_2	G	A	G	C	T	C
S_3	T	C	G	C	T	T

Sorted Suffixes of S′

$\$_1$
$\$_2$
$\$_3$
$AAC\$_1$
$AC\$_1$
$AGCTC\$_2$
$C\$_1$
$C\$_2$
$CAAC\$_1$
$CCAAC\$_1$
$CGCTT\$_3$
$CTC\$_2$
$CTT\$_3$
$GAGCTC\$_2$
$GCCAAC\$_1$
$GCTC\$_2$
$GCTT\$_3$
$T\$_3$
$TC\$_2$
$TCGCTT\$_3$
$TG\$_2$

- Sort all the suffixes of the strings in S′;
- Take the string $eBWT(S')$ obtained by concatenating the symbols that (circularly) precede the first symbol of each suffix in the list of (lexicographically) sorted suffixes of S′.

$$eBWT(S') = CCTCAGATCGTGG\$_2\$_1ACTC\$_3C$$

Note that the colors and suffixes are only for illustrative purposes.

How does EBWT work?

Let S $= \{S_1, S_2, \ldots, S_m\}$ be a collection of strings.

- Append an end-marker \$, where $\$_i < A < C < G < T$ (we add subscripts $\$_i$ only for illustrative purposes), to each string in S by obtaining a new collection S′.

String collection S′

	0	1	2	3	4	5	6
S_1	G	C	C	A	A	C	$\$_1$
S_2	G	A	G	C	T	C	$\$_2$
S_3	T	C	G	C	T	T	$\$_3$

Sorted Suffixes of S′

$\$_1$
$\$_2$
$\$_3$
$AAC\$_1$
$AC\$_1$
$AGCTC\$_2$
$C\$_1$
$C\$_2$
$CAAC\$_1$
$CCAAC\$_1$
$CGCTT\$_3$
$CTC\$_2$
$CTT\$_3$
$GAGCTC\$_2$
$GCCAAC\$_1$
$GCTC\$_2$
$GCTT\$_3$
$T\$_3$
$TC\$_2$
$TCGCTT\$_3$
$TG\$_2$

- Sort all the suffixes of the strings in S′;
- Take the string $eBWT(S')$ obtained by concatenating the symbols that (circularly) precede the first symbol of each suffix in the list of (lexicographically) sorted suffixes of S′.

$$eBWT(S') = CCTCAGATCGTGG\$_2\$_1ACTC\$_3C$$

Note that the colors and suffixes are only for illustrative purposes.

How does EBWT work?

Let $\mathsf{S} = \{S_1, S_2, \ldots, S_m\}$ be a collection of strings.

- Append an end-marker \$, where $\$_i < A < C < G < T$ (we add subscripts $\$_i$ only for illustrative purposes), to each string in S by obtaining a new collection S′.

String collection S′

	0	1	2	3	4	5	6
S_1	G	C	C	A	A	C	$\$_1$
S_2	G	A	G	C	T	C	$\$_2$
S_3	T	C	G	C	T	T	$\$_3$

- Sort all the suffixes of the strings in S′;
- Take the string $eBWT(\mathsf{S}')$ obtained by concatenating the symbols that (circularly) precede the first symbol of each suffix in the list of (lexicographically) sorted suffixes of S′.

$$eBWT(\mathsf{S}') = CCTCAGATCGTGG\$_2\$_1ACTC\$_3C$$

Sorted Suffixes of S′

$\$_1$
$\$_2$
$\$_3$
$AAC\$_1$
$AC\$_1$
$AGCTC\$_2$
$C\$_1$
$C\$_2$
$CAAC\$_1$
$CCAAC\$_1$
$CGCTT\$_3$
$CTC\$_2$
$CTT\$_3$
$GAGCTC\$_2$
$GCCAAC\$_1$
$GCTC\$_2$
$GCTT\$_3$
$T\$_3$
$TC\$_2$
$TCGCTT\$_3$
$TT\$_3$

Note that the colors and suffixes are only for illustrative purposes.

How does EBWT work?

Let $\mathsf{S} = \{S_1, S_2, \ldots, S_m\}$ be a collection of strings.

- Append an end-marker \$, where $\$_i < A < C < G < T$ (we add subscripts $\$_i$ only for illustrative purposes), to each string in S by obtaining a new collection S′.

String collection S′

	0	1	2	3	4	5	6
S_1	G	C	C	A	A	C	$\$_1$
S_2	G	A	G	C	T	C	$\$_2$
S_3	T	C	G	C	T	T	$\$_3$

- Sort all the suffixes of the strings in S′;
- Take the string $eBWT(\mathsf{S}')$ obtained by concatenating the symbols that (circularly) precede the first symbol of each suffix in the list of (lexicographically) sorted suffixes of S′.

$eBWT(\mathsf{S}')$	Sorted Suffixes of S′
C	$\$_1$
C	$\$_2$
T	$\$_3$
C	$AAC\$_1$
A	$AC\$_1$
G	$AGCTC\$_2$
A	$C\$_1$
T	$C\$_2$
C	$CAAC\$_1$
G	$CCAAC\$_1$
T	$CGCTT\$_3$
G	$CTC\$_2$
G	$CTT\$_3$
$\$_2$	$GAGCTC\$_2$
$\$_1$	$GCCAAC\$_1$
A	$GCTC\$_2$
C	$GCTT\$_3$
T	$T\$_3$
C	$TC\$_2$
$\$_3$	$TCGCTT\$_3$
C	$TTC\$_3$

$eBWT(\mathsf{S}') = CCTCAGATCGTGG\$_2\$_1ACTC\$_3C$

Note that the colors and suffixes are only for illustrative purposes.

How does EBWT work?

Let $\mathsf{S} = \{S_1, S_2, \ldots, S_m\}$ be a collection of strings.

- Append an end-marker \$, where $\$_i < A < C < G < T$ (we add subscripts $\$_i$ only for illustrative purposes), to each string in S by obtaining a new collection S′.

String collection S′

	0	1	2	3	4	5	6
S_1	G	C	C	A	A	C	$\$_1$
S_2	G	A	G	C	T	C	$\$_2$
S_3	T	C	G	C	T	T	$\$_3$

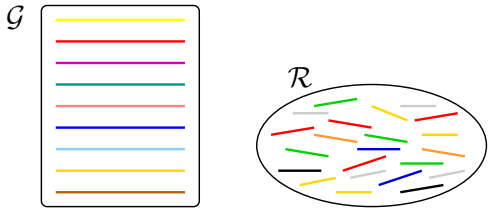
- Sort all the suffixes of the strings in S′;
- Take the string $eBWT(\mathsf{S}')$ obtained by concatenating the symbols that (circularly) precede the first symbol of each suffix in the list of (lexicographically) sorted suffixes of S′.

$eBWT(\mathsf{S}')$	Sorted Suffixes of S′
C	$\$_1$
C	$\$_2$
T	$\$_3$
C	$AAC\$_1$
A	$AC\$_1$
G	$AGCTC\$_2$
A	$C\$_1$
T	$C\$_2$
C	$CAAC\$_1$
G	$CCAAC\$_1$
T	$CGCTT\$_3$
G	$CTC\$_2$
G	$CTT\$_3$
$\$_2$	$GAGCTC\$_2$
$\$_1$	$GCCAAC\$_1$
A	$GCTC\$_2$
C	$GCTT\$_3$
T	$T\$_3$
C	$TC\$_2$
$\$_3$	$TCGCTT\$_3$
C	$TTC\$_3$

$eBWT(\mathsf{S}') = CCTCAGATCGTGG\$_2\$_1ACTC\$_3C$

Note that the colors and suffixes are only for illustrative purposes.

Formalization: Metagenomic classification problem



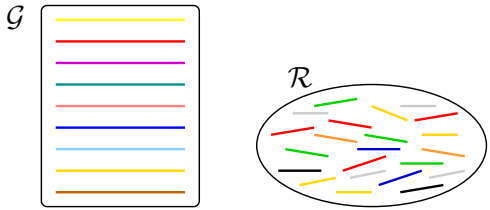
- $\mathcal{R} = \{r_1, \ldots, r_{|R|}\}$ metagenome (collection of short reads)
- $\mathcal{G} = \{g_1, \ldots, g_{|G|}\}$ reference genomes (collection of long sequences)

Pre-processing step: build the data structures $eBWT(\mathcal{S})$, $DA(\mathcal{S})$, $LCP(\mathcal{S})$, where $\mathcal{S} = \mathcal{R} \cup \mathcal{G}$.

The data structures for the reference database can be built only once, and updated with the data structures for the reads collection in order to obtain the data structures for $\mathcal{S}$.

Goal: to assign each read r_i in $\mathcal{R}$ to a unique genome g_j in $\mathcal{G}$

Formalization: Metagenomic classification problem



- $\mathcal{R} = \{r_1, \ldots, r_{|R|}\}$ metagenome (collection of short reads)
- $\mathcal{G} = \{g_1, \ldots, g_{|G|}\}$ reference genomes (collection of long sequences)

Pre-processing step: build the data structures $eBWT(\mathcal{S})$, $DA(\mathcal{S})$, $LCP(\mathcal{S})$, where $\mathcal{S} = \mathcal{R} \cup \mathcal{G}$.

The data structures for the reference database can be built only once, and updated with the data structures for the reads collection in order to obtain the data structures for $\mathcal{S}$.

Goal: to assign each read r_i in $\mathcal{R}$ to a unique genome g_j in $\mathcal{G}$

Preprocessing: eBWT, DA and LCP array

$DA[i]$: index of the string to which the i-th suffix belongs.
$LCP[i]$: length of the Longest Common Prefix between the i-th and the $(i-1)$-th suffix.

S = $\{GGCGTACCA\$_1, GGGGCGTAT\$_2, ACGATTAGC\$_3\}$

DA	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_1$
2	0	T	$\$_2$
3	0	C	$\$_3$
1	0	C	$A\$_1$
1	1	T	$ACCA\$_1$
3	2	$\$_3$	$ACGATTAGC\$_3$
3	1	T	$AGC\$_3$
2	1	T	$AT\$_2$
3	2	G	$ATTAGC\$_3$
3	0	G	$C\$_3$
1	1	C	$CA\$_1$
1	1	A	$CCA\$_1$
3	1	A	$CGATTAGC\$_3$
1	2	G	$CGTACCA\$_1$
2	4	G	$CGTAT\$_2$
3	0	C	$GATTAGC\$_3$
3	1	A	$GC\$_3$
1	2	G	$GCGTACCA\$_1$
2	5	G	$GCGTAT\$_2$
1	1	$\$_1$	$GGCGTACCA\$_1$
2	6	G	$GGCGTAT\$_2$
2	2	G	$GGGCGTAT\$_2$
2	3	$\$_2$	$GGGGCGTAT\$_2$
1	1	C	$GTACCA\$_1$
2	3	C	$GTAT\$_2$
2	0	A	$T\$_2$
1	1	G	$TACCA\$_1$
3	2	T	$TAGC\$_3$
2	2	G	$TAT\$_2$
3	1	A	$TTAGC\$_3$

Preprocessing: eBWT, DA and LCP array

$DA[i]$: index of the string to which the i-th suffix belongs.
$LCP[i]$: length of the Longest Common Prefix between the i-th and the $(i-1)$-th suffix.

S = $\{GGCGTACCA\$_1, GGGGCGTAT\$_2, ACGATTAGC\$_3\}$

DA	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_1$
2	0	T	$\$_2$
3	0	C	$\$_3$
1	0	C	$A\$_1$
1	1	T	$ACCA\$_1$
3	2	$\$_3$	$ACGATTAGC\$_3$
3	1	T	$AGC\$_3$
2	1	T	$AT\$_2$
3	2	G	$ATTAGC\$_3$
3	0	G	$C\$_3$
1	1	C	$CA\$_1$
1	1	A	$CCA\$_1$
3	1	A	$CGATTAGC\$_3$
1	2	G	$CGTACCA\$_1$
2	4	G	$CGTAT\$_2$

DA	LCP	$eBWT$	Sorted suffixes
3	0	C	$GATTAGC\$_3$
3	1	A	$GC\$_3$
1	2	G	$GCGTACCA\$_1$
2	5	G	$GCGTAT\$_2$
1	1	$\$_1$	$GGCGTACCA\$_1$
2	6	G	$GGCGTAT\$_2$
2	2	G	$GGGCGTAT\$_2$
2	3	$\$_2$	$GGGGCGTAT\$_2$
1	1	C	$GTACCA\$_1$
2	3	C	$GTAT\$_2$
2	0	A	$T\$_2$
1	1	G	$TACCA\$_1$
3	2	T	$TAGC\$_3$
2	2	G	$TAT\$_2$
3	1	A	$TTAGC\$_3$

Preprocessing: eBWT, DA and LCP array

$DA[i]$: index of the string to which the i-th suffix belongs.
$LCP[i]$: length of the Longest Common Prefix between the i-th and the $(i-1)$-th suffix.

S = $\{GGCGTACCA\$_1, GGGGCGTAT\$_2, ACGATTAGC\$_3\}$

DA	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_1$
2	0	T	$\$_2$
3	0	C	$\$_3$
1	0	C	$A\$_1$
1	1	T	$ACCA\$_1$
3	2	$\$_3$	$ACGATTAGC\$_3$
3	1	T	$AGC\$_3$
2	1	T	$AT\$_2$
3	2	G	$ATTAGC\$_3$
3	0	G	$C\$_3$
1	1	C	$CA\$_1$
1	1	A	$CCA\$_1$
3	1	A	$CGATTAGC\$_3$
1	2	G	$CGTACCA\$_1$
2	4	G	$CGTAT\$_2$
3	0	C	$GATTAGC\$_3$
3	1	A	$GC\$_3$
1	2	G	$GCGTACCA\$_1$
2	5	G	$GCGTAT\$_2$
1	1	$\$_1$	$GGCGTACCA\$_1$
2	6	G	$GGCGTAT\$_2$
2	2	G	$GGGCGTAT\$_2$
2	3	$\$_2$	$GGGGCGTAT\$_2$
1	1	C	$GTACCA\$_1$
2	3	C	$GTAT\$_2$
2	0	A	$T\$_2$
1	1	G	$TACCA\$_1$
3	2	T	$TAGC\$_3$
2	2	G	$TAT\$_2$
3	1	A	$TTAGC\$_3$

Preprocessing: eBWT, DA and LCP array

$DA[i]$: index of the string to which the i-th suffix belongs.
$LCP[i]$: length of the Longest Common Prefix between the i-th and the $(i-1)$-th suffix.

$S = \{GGCGTACCA\$_1, GGGGCGTAT\$_2, ACGATTAGC\$_3\}$

DA	LCP	$eBWT$
1	0	A
2	0	T
3	0	C
1	0	C
1	1	T
3	2	$\$_3$
3	1	T
2	1	T
3	2	G
3	0	G
1	1	C
1	1	A
3	1	A
1	2	G
2	4	G

DA	LCP	$eBWT$
3	0	C
3	1	A
1	2	G
2	5	G
1	1	$\$_1$
2	6	G
2	2	G
2	3	$\$_2$
1	1	C
2	3	C
2	0	A
1	1	G
3	2	T
2	2	G
3	1	A

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Property of the eBWT: Mixing and clustering effect

Intuitive idea

The greater is the number of substrings share by two strings, the smaller is their "distance"

Key property of the eBWT

The greater is the number of substrings shared by u and v, the greater is the mixing of the suffixes of u and v in the sorted list and the greater are the runs (clusters) of the same symbol in the eBWT.

u=$GGCGTACCA\$_1$. v=$GGGGCGTAT\$_2$

$eBWT$	Sorted suffixes
A	$\$_1$
T	$\$_2$
C	$A\$_1$
T	$ACCA\$_1$
T	$AT\$_2$
C	$CA\$_1$
A	$CCA\$_1$
G	$CGTACCA\$_1$
G	$CGTAT\$_2$
G	$GCGTACCA\$_1$
G	$GCGTAT\$_2$
$\$_1$	$GGCGTACCA\$_1$
G	$GGCGTAT\$_2$
G	$GGGCGTAT\$_2$
$\$_2$	$GGGGCGTAT\$_2$
C	$GTACCA\$_1$
C	$GTAT\$_2$
A	$T\$_2$
G	$TACCA\$_1$
G	$TAT\$_2$

Step 1: Build α-clusters and Similarity Arrays (Part I)

Minimum LCP value $\alpha = 3$

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

i	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_i$
2	0	T	$\$_j$
3	0	C	$A\$_i$
4	1	T	$ACCA\$_i$
5	1	T	$AGTTT\$_j$
6	1	T	$ATGTATTAGTTT\$_j$
7	2	T	$ATTAGTTT\$_j$
8	0	C	$CA\$_i$
9	1	A	$CCA\$_i$
10	1	G	$CGGGGCGTA\ldots\$_j$
11	2	G	$CGTACCA\$_i$
12	4	G	$CGTATGAT\ldots\$_j$
13	1	T	$CTTTTGGCG\ldots\$_j$
14	0	G	$GCGGGGCGT\ldots\$_j$
15	3	G	$GCGTACCA\$_i$
16	5	G	$GCGTATGTAA\ldots\$_j$
⋮	⋮	⋮	⋮

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[s] \in \mathcal{R}$ and $DA[t] \in \mathcal{G}$, $pS \leq s, t \leq pE$.

$\mathcal{C}_\alpha(r_i, g_j) = \{$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_{r_i}[g_j] =$

Step 1: Build α-clusters and Similarity Arrays (Part I)

Minimum LCP value $\alpha = 3$

i	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_i$
2	0	T	$\$_j$
3	0	C	$A\$_i$
4	1	T	$ACCA\$_i$
5	1	T	$AGTTT\$_j$
6	1	T	$ATGTATTAGTTT\$_j$
7	2	T	$ATTAGTTT\$_j$
8	0	C	$CA\$_i$
9	1	A	$CCA\$_i$
10	1	G	$CGGGGCGTA\ldots\$_j$
11	2	G	$CGTACCA\$_i$
12	4	G	$CGTATGAT\ldots\$_j$
13	1	T	$CTTTTGGCG\ldots\$_j$
14	0	G	$GCGGGGCGT\ldots\$_j$
15	3	G	$GCGTACCA\$_i$
16	5	G	$GCGTATGTAA\ldots\$_j$
⋮	⋮	⋮	⋮

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[s] \in \mathcal{R}$ and $DA[t] \in \mathcal{G}$, $pS \leq s, t \leq pE$.

$\mathcal{C}_\alpha(r_i, g_j) = \{(11, 12),$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_{r_i}[g_j] = 1+$

Step 1: Build α-clusters and Similarity Arrays (Part I)

Minimum LCP value $\alpha = 3$

i	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_i$
2	0	T	$\$_j$
3	0	C	$A\$_i$
4	1	T	$ACCA\$_i$
5	1	T	$AGTTT\$_j$
6	1	T	$ATGTATTAGTTT\$_j$
7	2	T	$ATTAGTTT\$_j$
8	0	C	$CA\$_i$
9	1	A	$CCA\$_i$
10	1	G	$CGGGGCGTA\ldots\$_j$
11	2	G	$CGTACCA\$_i$
12	4	G	$CGTATGAT\ldots\$_j$
13	1	T	$CTTTTGGCG\ldots\$_j$
14	0	G	$GCGGGGGCGT\ldots\$_j$
15	3	G	$GCGTACCA\$_i$
16	5	G	$GCGTATGTAA\ldots\$_j$
⋮	⋮	⋮	⋮

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[s] \in \mathcal{R}$ and $DA[t] \in \mathcal{G}$, $pS \leq s, t \leq pE$.

$\mathcal{C}_\alpha(r_i, g_j) = \{(11,12),(14,16),\ldots$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_{r_i}[g_j] = 1{+}1{+}$

Step 1: Build α-clusters and Similarity Arrays (Part I)

Minimum LCP value $\alpha = 3$

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

i	LCP	$eBWT$	Sorted suffixes
1	0	A	$\$_i$
2	0	T	$\$_j$
3	0	C	$A\$_i$
4	1	T	$ACCA\$_i$
5	1	T	$AGTTT\$_j$
6	1	T	$ATGTATTAGTTT\$_j$
7	2	T	$ATTAGTTT\$_j$
8	0	C	$CA\$_i$
9	1	A	$CCA\$_i$
10	1	G	$CGGGGCGTA\ldots\$_j$
11	2	G	$CGTACCA\$_i$
12	4	G	$CGTATGAT\ldots\$_j$
13	1	T	$CTTTTGGCG\ldots\$_j$
14	0	G	$GCGGGGCGT\ldots\$_j$
15	3	G	$GCGTACCA\$_i$
16	5	G	$GCGTATGTAA\ldots\$_j$
⋮	⋮	⋮	⋮

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[s] \in \mathcal{R}$ and $DA[t] \in \mathcal{G}$, $pS \leq s, t \leq pE$.

$\mathcal{C}_\alpha(r_i, g_j) = \{(11,12),(14,16),\ldots$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_{r_i}[g_j] = 1{+}1{+}\ldots$

Step 1: Build α-clusters and Similarity Arrays (Part II)

Minimum LCP value $\alpha = 3$

i	LCP	$eBWT$	Sorted suffixes
⋮	⋮	⋮	⋮
17	1	T	$GGCGGGGCG\ldots\$_j$
18	4	K	$GGCGTACCA\$_i$
19	6	G	$GGCGTATGTAT\ldots\$_j$
20	2	G	$GGGCGTAT\ldots\$_j$
21	3	C	$GGGGCGTAT\ldots\$_j$
22	1	C	$GTACCA\$_i$
23	3	C	$GTATGTA\ldots\$_j$
24	4	C	$GTATTA\ldots\$_j$
25	2	A	$GTTT\$_j$
26	0	$\$_i$	$KGGCGTACCA\$_i$
27	0	T	$T\$_j$
28	1	G	$TACCA\$_i$
29	2	T	$TAGTTT\$_j$
⋮	⋮	⋮	⋮

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[i] \in \mathcal{R}$ and $DA[j] \in \mathcal{G}$, $pS \leq i, j \leq pE$.

$\mathcal{C}_\alpha = \{(11, 12), (14, 16),$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_r[g] = 1{+}1{+}$

Step 1: Build α-clusters and Similarity Arrays (Part II)

Minimum LCP value $\alpha = 3$

i	LCP	$eBWT$	Sorted suffixes
⋮	⋮	⋮	⋮
17	1	T	$GGCGGGGCG\ldots\$_j$
18	4	K	$GGCGTACCA\$_i$
19	6	G	$GGCGTATGTAT\ldots\$_j$
20	2	G	$GGGCGTAT\ldots\$_j$
21	3	C	$GGGGCGTAT\ldots\$_j$
22	1	C	$GTACCA\$_i$
23	3	C	$GTATGTA\ldots\$_j$
24	4	C	$GTATTA\ldots\$_j$
25	2	A	$GTTT\$_j$
26	0	$\$_i$	$KGGCGTACCA\$_i$
27	0	T	$T\$_j$
28	1	G	$TACCA\$_i$
29	2	T	$TAGTTT\$_j$
⋮	⋮	⋮	⋮

$K \rightarrow \{G \text{ or } T\}$ in the IUPAC list.

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[i] \in \mathcal{R}$ and $DA[j] \in \mathcal{G}$, $pS \leq i, j \leq pE$.

$\mathcal{C}_\alpha = \{(11,12),(14,16),(17,19),$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_r[g] = 1{+}1{+}1{+}$

Step 1: Build α-clusters and Similarity Arrays (Part II)

Minimum LCP value $\alpha = 3$

i	LCP	$eBWT$	Sorted suffixes
⋮	⋮	⋮	⋮
17	1	T	$GGCGGGGCG\ldots\$_j$
18	4	K	$GGCGTACCA\$_i$
19	6	G	$GGCGTATGTAT\ldots\$_j$
20	2	G	$GGGCGTAT\ldots\$_j$
21	3	C	$GGGGCGTAT\ldots\$_j$
22	1	C	$GTACCA\$_i$
23	3	C	$GTATGTA\ldots\$_j$
24	4	C	$GTATTA\ldots\$_j$
25	2	A	$GTTT\$_j$
26	0	$\$_i$	$KGGCGTACCA\$_i$
27	0	T	$T\$_j$
28	1	G	$TACCA\$_i$
29	2	T	$TAGTTT\$_j$
⋮	⋮	⋮	⋮

it is not cluster (only blue symbols)

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An α-*cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[i] \in \mathcal{R}$ and $DA[j] \in \mathcal{G}$, $pS \leq i, j \leq pE$.

$\mathcal{C}_\alpha = \{(11,12),(14,16),(17,19),$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_r[g] = 1{+}1{+}1{+}$

Step 1: Build α-clusters and Similarity Arrays (Part II)

Minimum LCP value $\alpha = 3$

i	LCP	$eBWT$	Sorted suffixes
⋮	⋮	⋮	⋮
17	1	T	$GGCGGGGCG\ldots\$_j$
18	4	K	$GGCGTACCA\$_i$
19	6	G	$GGCGTATGTAT\ldots\$_j$
20	2	G	$GGGCGTAT\ldots\$_j$
21	3	C	$GGGGCGTAT\ldots\$_j$
22	1	C	$GTACCA\$_i$
23	3	C	$GTATGTA\ldots\$_j$
24	4	C	$GTATTA\ldots\$_j$
25	2	A	$GTTT\$_j$
26	0	$\$_i$	$KGGCGTACCA\$_i$
27	0	T	$T\$_j$
28	1	G	$TACCA\$_i$
29	2	T	$TAGTTT\$_j$
⋮	⋮	⋮	⋮

r_i=$KGGCGTACCA\$_i$

g_j=$TTATTTTGGCGGGGCGTATGTATTAGTTT\$_j$

Detect blocks (α-clusters) of $eBWT(\mathcal{S})$: An *α-cluster* $\mathcal{C}_\alpha$ of $eBWT(\mathcal{S})$ is any pair of indices (pS, pE) such that

- $LCP[pS] < \alpha$ and $LCP[pE+1] < \alpha$,
- $LCP[k] \geq \alpha$, $pS < k \leq pE$,
- $DA[i] \in \mathcal{R}$ and $DA[j] \in \mathcal{G}$, $pS \leq i, j \leq pE$.

$\mathcal{C}_\alpha = \{(11,12),(14,16),(17,19),(22,24)\}$

Compute the similarity between any read and any genome:

$$Sim_r[g] = \sum_{x \in \mathcal{C}_\alpha} \sum_{a \in \Sigma} \min(n_r, n_g)$$

n_r=number of indices s in x such that $eBWT[s] = a$ and $DA[s] = r$,
n_g=number of indices t in x such that $eBWT[t] = a$ and $DA[t] = g$.
i.e. the total number of bases of r and g that match (by using IUPAC list) in each α-cluster.

$Sim_r[g] = 1{+}1{+}1{+}1 = 4$

Step 2: Classification

Given a threshold value β, the read r_i is

- assigned to g_j if g_j is the only genome such that $Sim_{r_i}[g_j] \sim \max_g Sim_{r_i}[g]$ and $Sim_{r_i}[g_j] > \beta$.
- not classified if $\max_g Sim_{r_i}[g] \leq \beta$.
- ambiguous if $\max_g Sim_{r_i}[g] > \beta$, but there exist at least two genomes g_p and g_q s.t. $Sim_{r_i}[g_p] \sim Sim_{r_i}[g_q] \sim \max_g Sim_{r_i}[g]$

Example

Let $\alpha = 3$ and $\beta = 0.4$.

Suppose the α-similarity between r_i and g_1, g_2, g_3, g_4, g_5 is

$Sim_{r_i}[g_1] = 0.5$, $Sim_{r_i}[g_2] = 0$, $Sim_{r_i}[g_3] =$, $Sim_{r_i}[g_4] = 0.2$, $Sim_{r_i}[g_5] = 0$.

Step 2: Classification

Given a threshold value β, the read r_i is

- assigned to g_j if g_j is the only genome such that $Sim_{r_i}[g_j] \sim \max_g Sim_{r_i}[g]$ and $Sim_{r_i}[g_j] > \beta$.
- not classified if $\max_g Sim_{r_i}[g] \leq \beta$.
- ambiguous if $\max_g Sim_{r_i}[g] > \beta$, but there exist at least two genomes g_p and g_q s.t. $Sim_{r_i}[g_p] \sim Sim_{r_i}[g_q] \sim \max_g Sim_{r_i}[g]$

Example

Let $\alpha = 3$ and $\beta = 0.4$.
Suppose the α-similarity between r_i and g_1, g_2, g_3, g_4, g_5 is
$Sim_{r_i}[g_1] = 0.5$, $Sim_{r_i}[g_2] = 0$, $Sim_{r_i}[g_3] = 0.8$, $Sim_{r_i}[g_4] = 0.2$, $Sim_{r_i}[g_5] = 0$.

Step 2: Classification

Given a threshold value β, the read r_i is

- assigned to g_j if g_j is the only genome such that $Sim_{r_i}[g_j] \sim \max_g Sim_{r_i}[g]$ and $Sim_{r_i}[g_j] > \beta$.
- not classified if $\max_g Sim_{r_i}[g] \leq \beta$.
- ambiguous if $\max_g Sim_{r_i}[g] > \beta$, but there exist at least two genomes g_p and g_q s.t. $Sim_{r_i}[g_p] \sim Sim_{r_i}[g_q] \sim \max_g Sim_{r_i}[g]$

Example

Let $\alpha = 3$ and $\beta = 0.4$.
Suppose the α-similarity between r_i and g_1, g_2, g_3, g_4, g_5 is
$Sim_{r_i}[g_1] = 0.5$, $Sim_{r_i}[g_2] = 0$, $Sim_{r_i}[g_3] = 0.8$, $Sim_{r_i}[g_4] = 0.2$, $Sim_{r_i}[g_5] = 0$.
$\Rightarrow r_i$ is assigned to g_3.

Step 2: Classification

Given a threshold value β, the read r_i is

- assigned to g_j if g_j is the only genome such that $Sim_{r_i}[g_j] \sim \max_g Sim_{r_i}[g]$ and $Sim_{r_i}[g_j] > \beta$.
- not classified if $\max_g Sim_{r_i}[g] \leq \beta$.
- ambiguous if $\max_g Sim_{r_i}[g] > \beta$, but there exist at least two genomes g_p and g_q s.t. $Sim_{r_i}[g_p] \sim Sim_{r_i}[g_q] \sim \max_g Sim_{r_i}[g]$

Example

Let $\alpha = 3$ and $\beta = 0.4$.
Suppose the α-similarity between r_i and g_1, g_2, g_3, g_4, g_5 is
$Sim_{r_i}[g_1] = 0.3$, $Sim_{r_i}[g_2] = 0$, $Sim_{r_i}[g_3] = 0.34$, $Sim_{r_i}[g_4] = 0.2$, $Sim_{r_i}[g_5] = 0$.

Step 2: Classification

Given a threshold value β, the read r_i is

- assigned to g_j if g_j is the only genome such that $Sim_{r_i}[g_j] \sim \max_g Sim_{r_i}[g]$ and $Sim_{r_i}[g_j] > \beta$.
- not classified if $\max_g Sim_{r_i}[g] \leq \beta$.
- ambiguous if $\max_g Sim_{r_i}[g] > \beta$, but there exist at least two genomes g_p and g_q s.t. $Sim_{r_i}[g_p] \sim Sim_{r_i}[g_q] \sim \max_g Sim_{r_i}[g]$

Example

Let $\alpha = 3$ and $\beta = 0.4$.
Suppose the α-similarity between r_i and g_1, g_2, g_3, g_4, g_5 is
$Sim_{r_i}[g_1] = 0.5$, $Sim_{r_i}[g_2] = 0$, $Sim_{r_i}[g_3] = 0.5$, $Sim_{r_i}[g_4] = 0.2$, $Sim_{r_i}[g_5] = 0$.
$\Rightarrow r_i$ is ambiguous.

Preliminary experiments on simulated reads

- Positive and negative control datasets (about 20 millions of paired-reads) designed in [Lindgreen et al., 2016].
- Reference database $\mathcal{G}$: 930 genomes from 686 species

setA2	CLARK-S –highconf	LightMetaEbwt α 16 β 0.25	LightMetaEbwt α 16 β 0.35	Centrifuge -min-hitlen 16	Centrifuge -min-hitlen 22
SEN (%)	93.03	**93.13**	92.69	**95.65**	93.01
PREC (%)	99.06	99.73	**99.74**	97.64	99.66
F1 (%)	95.95	**96.32**	96.08	**96.63**	96.22
setB2					
SEN (%)	92.84	**93.01**	92.51	**95.53**	92.94
PREC (%)	99.11	99.77	**99.78**	97.68	99.69
F1 (%)	95.87	**96.27**	96.01	**96.59**	96.20
setA2Ran					
TN	5,726,336	5,726,294	5,726,357	150,971	5,712,085
FP	22	64	1	5,575,387	14,273
SPEC (%)	99.99	99.99	**100.00**	**2.64**	99.75
setB2Ran					
TN	5,406,642	5,406,601	5,406,658	141,994	5,393,260
FP	17	58	1	5,264,665	13,399
SPEC (%)	99.99	99.99	**100.00**	**2.63**	99.75

SEN = proportion of the actual positives identified by the method.
PREC = proportion of positives that are correctely identified by the method.
F1 = harmonic mean between **SEN** and **PREC**.
SPEC = proportion of actual negatives that are correctely identified as such.

Preliminary experiments on simulated reads

- Positive and negative control datasets (about 20 millions of paired-reads) designed in [Lindgreen et al., 2016].
- Reference database $\mathcal{G}$: 930 genomes from 686 species

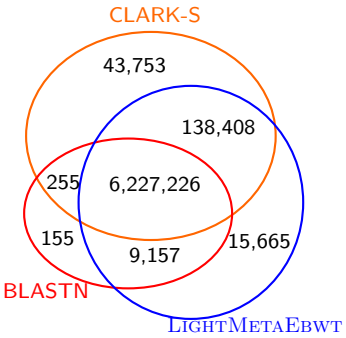
setA2	CLARK-S –highconf	LightMetaEbwt α 16 β 0.25	LightMetaEbwt α 16 β 0.35	Centrifuge -min-hitlen 16	Centrifuge -min-hitlen 22
SEN (%)	93.03	**93.13**	92.69	**95.65**	93.01
PREC (%)	99.06	99.73	**99.74**	97.64	99.66
F1 (%)	95.95	**96.32**	96.08	**96.63**	96.22
setB2					
SEN (%)	92.84	**93.01**	92.51	**95.53**	92.94
PREC (%)	99.11	99.77	**99.78**	97.68	99.69
F1 (%)	95.87	**96.27**	96.01	**96.59**	96.20
setA2Ran					
TN	5,726,336	5,726,294	5,726,357	150,971	5,712,085
FP	22	64	1	5,575,387	14,273
SPEC (%)	99.99	99.99	**100.00**	**2.64**	99.75
setB2Ran					
TN	5,406,642	5,406,601	5,406,658	141,994	5,393,260
FP	17	58	1	5,264,665	13,399
SPEC (%)	99.99	99.99	**100.00**	**2.63**	99.75

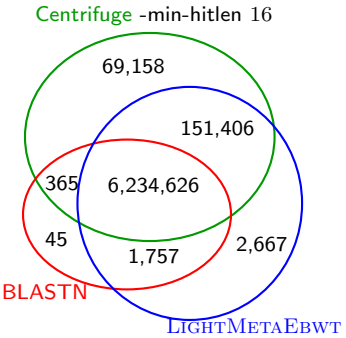
SEN = proportion of the actual positives identified by the method.
PREC = proportion of positives that are correctely identified by the method.
F1 = harmonic mean between **SEN** and **PREC**.
SPEC = proportion of actual negatives that are correctely identified as such.

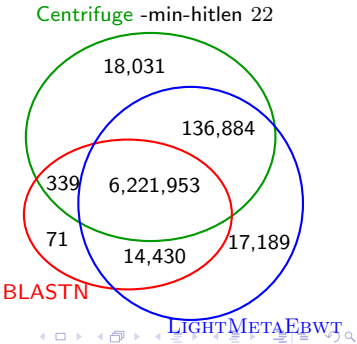
Preliminary experiments on real data: Mock community

Reference database $\mathcal{G}$: 61 genomes from 22 species

SRR172902	BLASTN -eval 10^{-5} -id 90	CLARK-S -c 0.75 -g 0.08	LightMetaEbwt α 15 β 0.2	Centrifuge -min-hitlen 16	Centrifuge -min-hitlen 22
Classified	6,236,793	6,409,642	6,390,456	6,455,555	6,377,207
Ambiguous	48,310	0	10,570	11,150	8,422
Unclassified	276,962	152,423	161,039	95,360	176,436
SEN (%)	95.04	97.68	97.39	98.38	97.18



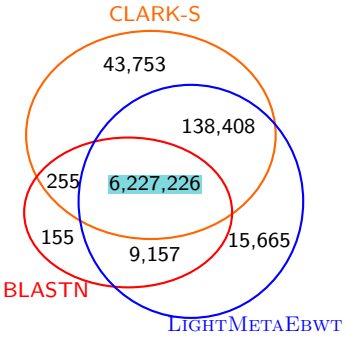


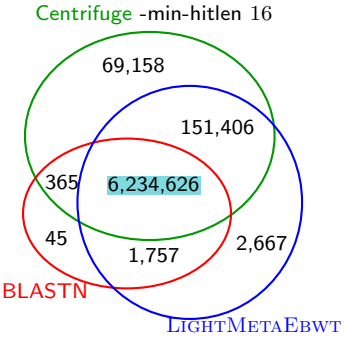


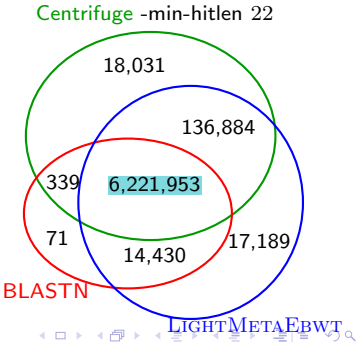
Preliminary experiments on real data: Mock community

Reference database $\mathcal{G}$: 61 genomes from 22 species

SRR172902	BLASTN -eval 10^{-5} -id 90	CLARK-S -c 0.75 -g 0.08	LightMetaEbwt α 15 β 0.2	Centrifuge -min-hitlen 16	Centrifuge -min-hitlen 22
Classified	6,236,793	6,409,642	6,390,456	6,455,555	6,377,207
Ambiguous	48,310	0	10,570	11,150	8,422
Unclassified	276,962	152,423	161,039	95,360	176,436
SEN (%)	95.04	97.68	97.39	98.38	97.18



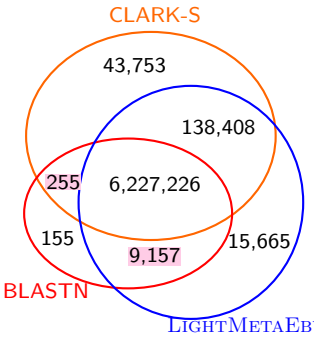


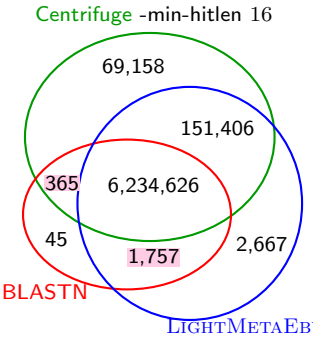


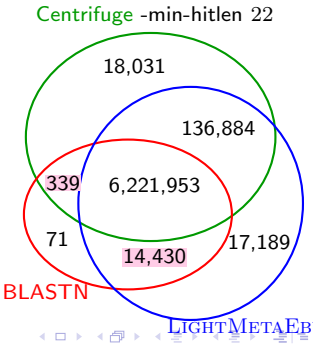
Preliminary experiments on real data: Mock community

Reference database $\mathcal{G}$: 61 genomes from 22 species

SRR172902	BLASTN -eval 10^{-5} -id 90	CLARK-S -c 0.75 -g 0.08	LightMetaEbwt α 15 β 0.2	Centrifuge -min-hitlen 16	Centrifuge -min-hitlen 22
Classified	6,236,793	6,409,642	6,390,456	6,455,555	6,377,207
Ambiguous	48,310	0	10,570	11,150	8,422
Unclassified	276,962	152,423	161,039	95,360	176,436
SEN (%)	95.04	97.68	97.39	98.38	97.18







Final Remarks

Notice that a like-for-like comparison on the time-consuming and on memory-consuming between LightMetaEbwt, CLARK-S and Cenfrifuge is not possible

LightMetaEbwt	Centrifuge	Clark-S
One-core	multi-thread	multi-thread
Analyze all paired-reads at the same time, this implies keeping in memory the whole similarity matrix	Analyze one read at a time and Pair-reads analyzed in parallel	Analyze one read at a time and Concatenate paired-reads
no engineered implementation	engineered implementation	engineered implementation

	Classification for setA2 (21,461,160 paired-reads and 930 genomes)			
	LightMetaEbwt	Centrifuge	CLARK-S	
setA2	α 16	-min-hitlen 16	–highconf	last step
RAM	$\sim$ 1GB ($+\sim$ 18GB for the matrix)	$\sim$ 2GB	$\sim$ 121GB	$\sim$ 78GB
Time	$\sim$ 54 min	$\sim$ 27min	$\sim$ 8h	$\sim$ 28min

Here, we improved our previous version of this method presented to AICoB 2019: for instance, considering the IUPAC list. We are now working on a more engineered implementation ...

Thank you!

Preliminary experiments

Dataset	$\|\mathcal{R}\|$	$\|\mathcal{G}\|$	Instance	Efficiency	Wall Clock	RAM
setA2	21,461,160	930	CLARK-S (8cores)	99%	8:25:17	121.58
			LightMetaEbwt	99%	2:29:57	19.45
			Centrifuge	99%	1:55:10+26:38	9.32
setB2	20,249,373	930	CLARK-S (8cores)	69%	14:26:31	121.58
			LightMetaEbwt	96%	3:06:05	18.55
			Centrifuge	99%	1:55:10+21:16	9.32
setA2Ran	5,726,358	930	CLARK-S (8cores)	92%	5:02:22	121.58
			LightMetaEbwt	98%	18:46	9.94
			Centrifuge	99%	1:55:10+4:08	9.32
setB2Ran	5,406,659	930	CLARK-S (8cores)	99%	5:16:50	121.58
			LightMetaEbwt	99%	17:01	9.61
			Centrifuge	99%	1:55:10+3:20	9.32
SRR172902	6,562,065	61	CLARK-S (8cores)	99%	33:40	72.23
			LightMetaEbwt	99%	5:57	0.69
			Centrifuge	99%	1:21+2:29	0.37

- The three data structures for LightMetaEbwt setA2 take 1:23:55 and 4GB of RAM to be constructed.
- For LightMetaEbwt, post-processing time and space usage differ from step to step. For instance, per each set of paired end read in setA2 step 1 takes 1:15 with 1GB of RAM (changeable), step 2 around 36:17 with 19.45GB (18.59GB to keep the matrix of similarities) and step 3 takes 15:55 with 0.06GB.