CS4330: Combinatorial Methods in Bioinformatics

Read error correction using K-mers

Wong Limsoon

Acknowledgement: This set of slides were adapted from Ken Sung's

NUS
National University of Singapore
National University of Singapore

Errors in reads greatly increase complexity of genome assembly

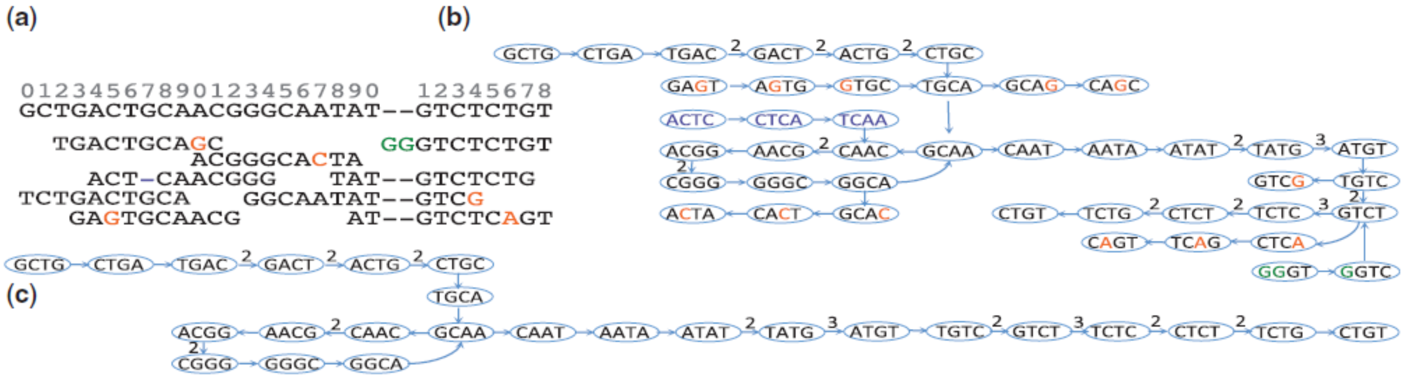


Fig. 1 An example of NGS data and its de Bruijn graph. The short stretches of sequences in (**a**) are the reads generated from an NGS platform, while the long sequence is the reference. The reference is often unknown but, for ease of illustration, it is shown here to demonstrate substitutions (coloured in orange), insertions (green) or deletions (light blue) errors. There is no '-' in the real-life reference and sequenced reads, but it is shown here also for better understanding. (**b**) The de Bruijn graph constructed from all the short sequences in (a) with a *k*-mer size of 4. (c) is the simplified error-corrected version of the de Bruijn graph of (b). The numbers along the edges represent their multiplicities

The error-containing de Bruijn graph (b) is much more complicated than the error-free graph (c)

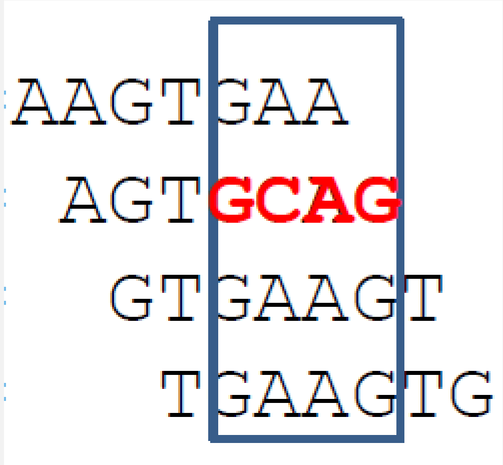
L. Zhao et al., "MapReduce for accurate error correction of next-generation sequencing data", *Bioinformatics* 33(23):3844-3851, 2017

Reads containing low-freq K-mers are much more likely to have errors

When a genome is sampled at high coverage, any K-mer in the genome can be expected to appear in many reads

For any K-mer t, let freq(t) = # of reads containing t or its reverse complement

If freq(t) is small, it is likely that some error has occurred in the reads containing t



Exercise

Reads containing low-frequency K-mers are likely to contain sequencing errors

Reads with errors greatly increase complexity of genome assembly

We should discard these reads and not use them in genome assembly, no?

Solid K-mers

A K-mer t is said to be solid wrt a set of sequencing reads $\mathcal{R}$ if freq(t) > M, where M is a given threshold

Solid K-mers are considered reliable due to their high frequency within the set of sequencing reads

Example

Read set, $\mathcal{R}$

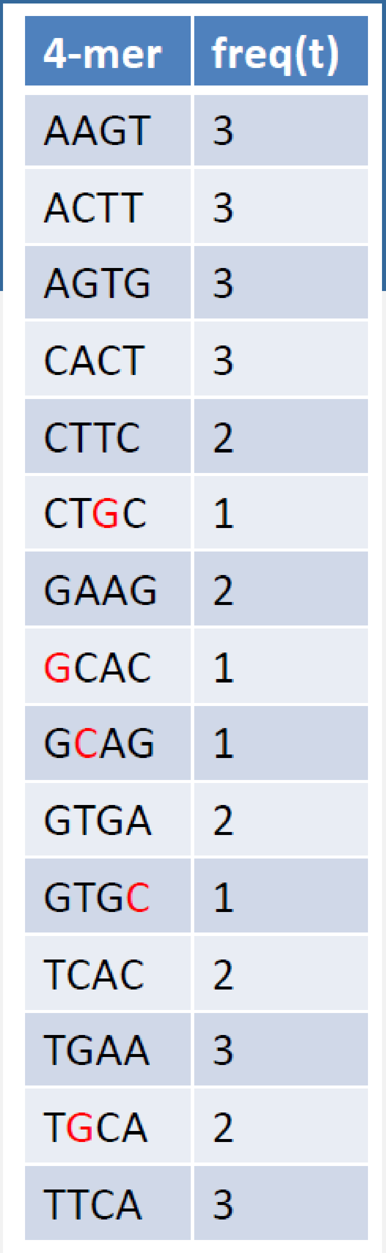
```
AAGTGAA
 AGTGCAG
  GTGAAGT
   TGAAGTG
```

K-mer t is solid if freq(t) > M

E.g., M = 2, the solid K-mers are:

AAGT, **ACTT**, AGTG

CACT, TGAA, **TTCA**



4-mer	freq(t)
AAGT	3
ACTT	3
AGTG	3
CACT	3
CTTC	2
CTGC	1
GAAG	2
GCAC	1
GCAG	1
GTGA	2
GTGC	1
TCAC	2
TGAA	3
TGCA	2
TTCA	3

The read error correction problem

Given a set of reads $\mathcal{R}$

Let $\mathcal{T}$ = the set of all correct K-mers in the genome

$\mathcal{T}$ is often approximated by solid K-mers in $\mathcal{R}$ in practice

A read R is a $\mathcal{T}$-string if every K-mer in R is in $\mathcal{T}$

Objective: Convert every read R $\in \mathcal{R}$ to R' by the minimum # of mutations such that R' is a $\mathcal{T}$-string

Exercise

Read set, $\mathcal{R}$

AAGTGAA
 AGTGCAG
  GTGAAGT
   TGAAGTG

$\mathcal{T}$ = solid K-mers, freq>1

AAGT, **ACTT**, AGTG
CACT, TGAA, **TTCA**,
CTTC, GAAG, GTGA,
TCAC, **TGCA**

Which reads in $\mathcal{R}$ are $\mathcal{T}$-strings?

Can you convert the non $\mathcal{T}$-string reads to $\mathcal{T}$-strings using min # of mutations?

Exercise

Read set, $\mathcal{R}$

```
AAGTGAA
 AGTGCAG
  GTGAAGT
   TGAAGTG
```

$\mathcal{T}$ = solid K-mers, freq>2

AAGT, **ACTT**, AGTG

CACT, TGAA, **TTCA**

Which reads in $\mathcal{R}$ are $\mathcal{T}$-strings?

Can you convert the non $\mathcal{T}$-string reads to $\mathcal{T}$-strings using min # of mutations?

Recursive spectra alignment

For a read R, find

$\min_{t \in \mathcal{T}} \text{dist}(|R|,t)$ where dist(i,t) = minimum edit distance betw R[1..i] and any $\mathcal{T}$-string that ends at K-mer t

Assume no indel error in first k bases of R

Let $\rho(x,y) = 0$ if $x = y$ and $\rho(x,y) = 1$ if $x \neq y$

Base case, i = K:

$$\text{dist}(K,t) = \begin{cases} \text{Hamming}(R[1..K],t) \text{ if } t \in \mathcal{T} \\ \infty \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

Recurrence:

$$\text{dist}(i,t) = \min \begin{cases} \min_{b \in \{A,C,G,T\}} \text{dist}(i-1, \boxed{b \bullet t[1..K-1]}) + \rho(R[i],t[K]) & \text{match} \\ \text{dist}(i-1, t) + 1 & \text{delete} \\ \min_{b \in \{A,C,G,T\}} \text{dist}(i, \boxed{b \bullet t[1..K-1]}) + 1 & \text{insert} \end{cases}$$

b • t[1..K-1]) must be solid K-mer

Potential infinite loop!

Chaisson et al., *Bioinformatics* 20(13):2067-2074, 2004

The “dependency graph” is cyclic but non-negative

R = AGTGCAG

T = { AAGT, AGTG,
GAAG, GTGA,
TGAA, TGCA }

Recurrence:

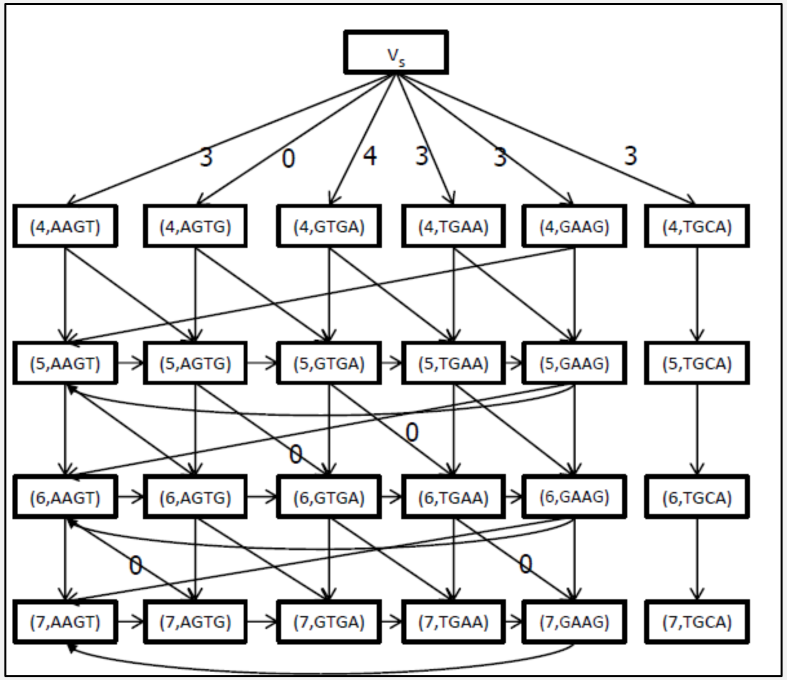
$$\text{dist}(i,t) = \min \begin{cases} \min_{b \in \{A,C,G,T\}} \text{dist}(i-1, b \bullet t[1..K-1]) + \rho(R[i],t[K]) & \text{match} \\ \text{dist}(i-1, t) + 1 & \text{delete} \\ \min_{b \in \{A,C,G,T\}} \text{dist}(i, b \bullet t[1..K-1]) + 1 & \text{insert} \end{cases}$$

R[1..4] = AGTG

R[5] = C

R[6] = A

R[7] = G



(mis)match = slant edge

delete = vertical edge

insert = horizontal edge

Spectra alignment via “shortest path” of dependency graph

Key lemma

dist(i,t) = length of shortest path from v_s *to (i,t)*

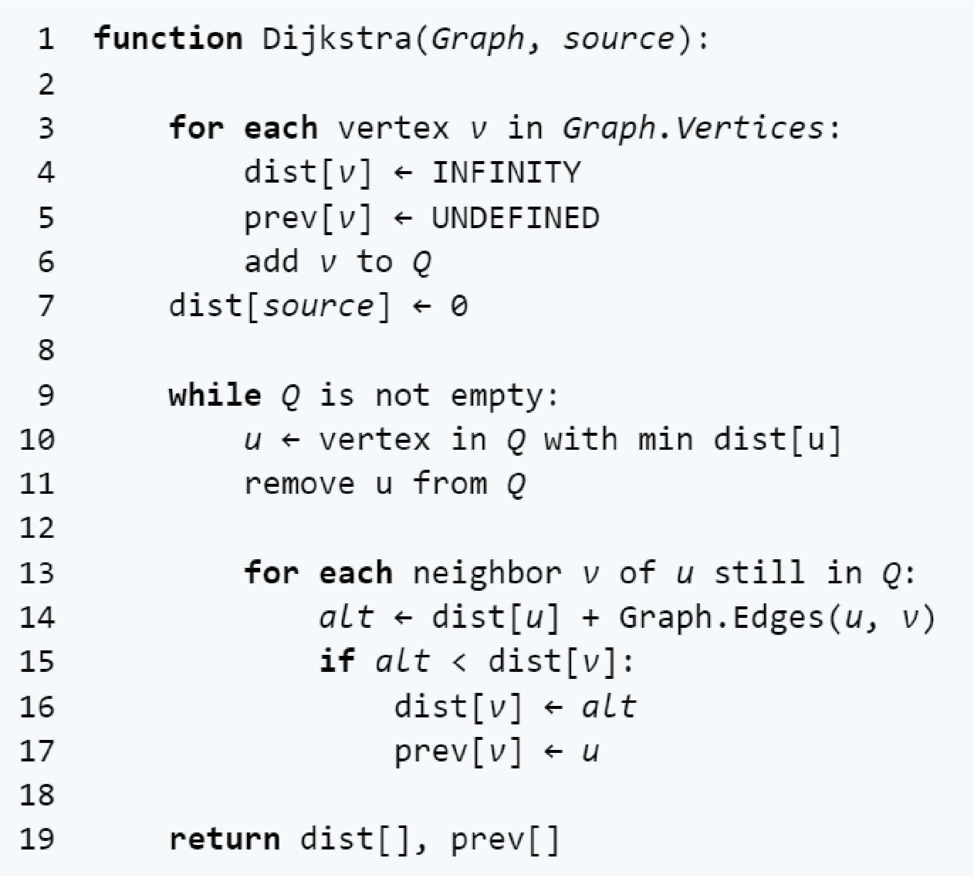
$\therefore$ Construct dependency graph; find shortest path from v_s to $(|R|,t)$ for some $t \in \mathcal{T}$

The dependency graph has $O(|R|\ |\mathcal{T}|)$ nodes and edges

$\therefore$ Complexity of graph construction = $O(|R|\ |\mathcal{T}|)$

$\therefore$ Complexity of shortest path finding = $O(|R|\ |\mathcal{T}|)$

Dijkstra’s shortest path algorithm



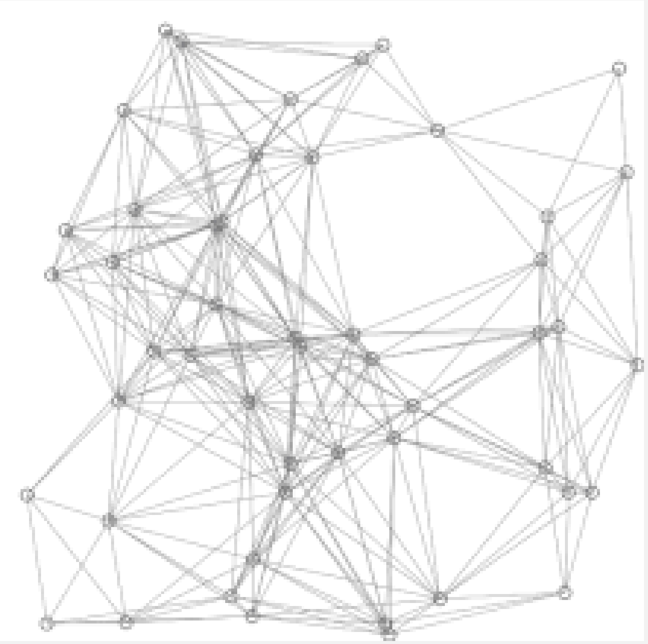
```
function Dijkstra(Graph, source):

    for each vertex v in Graph.Vertices:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]
        remove u from Q

        for each neighbor v of u still in Q:
            alt ← dist[u] + Graph.Edges(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```



Source: Wikipedia

Example

R = AGTGCAG

T = { AAGT, AGTG,
GAAG, GTGA,
TGAA, TGCA }

Min path length = 1

Corrected read = AGTGAAG

Recurrence:

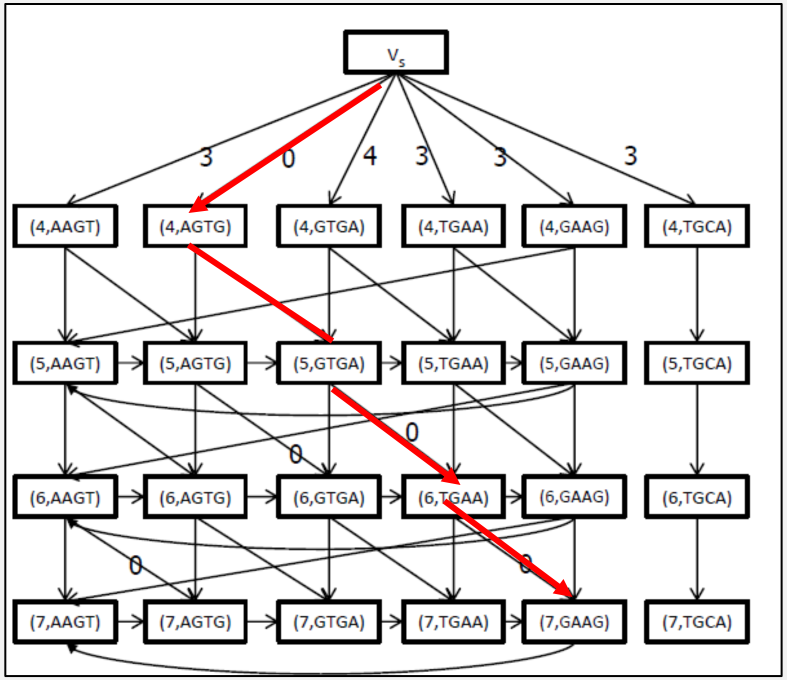
$$\mathrm{dist}(i,t) = \min \begin{cases} \min_{b \in \{A,C,G,T\}} \mathrm{dist}(i-1, b \bullet t[1..K-1]) + \rho(R[i],t[K]) & \text{match} \\ \mathrm{dist}(i-1, t) + 1 & \text{delete} \\ \min_{b \in \{A,C,G,T\}} \mathrm{dist}(i, b \bullet t[1..K-1]) + 1 & \text{insert} \end{cases}$$

R[1..4] = AGTG

R[5] = C

R[6] = A

R[7] = G

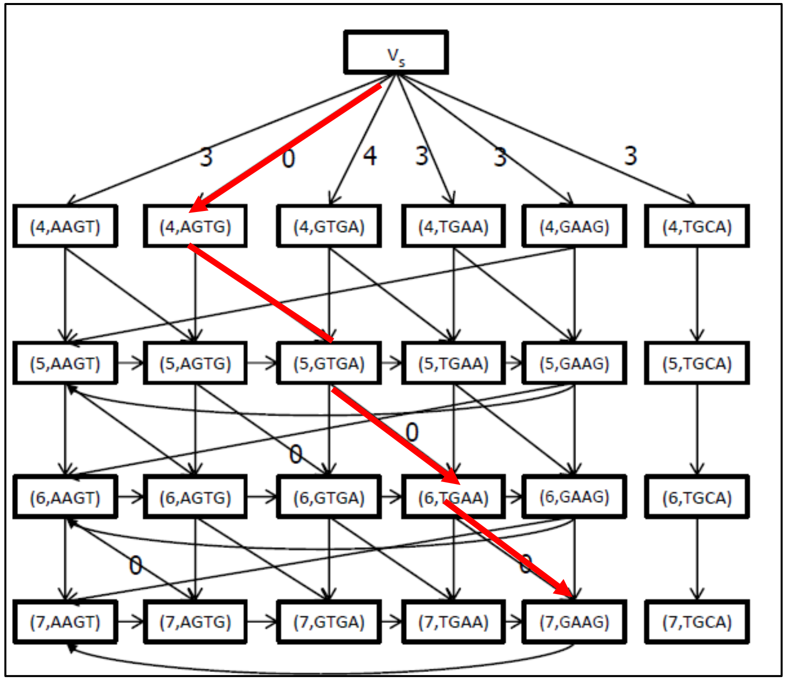


Exercise

Recurrence:

$$\text{dist}(i,t) = \min \begin{cases} \min_{b \in \{A,C,G,T\}} \text{dist}(i-1, b \bullet t[1..K-1]) + \rho(R[i],t[K]) & \text{match} \\ \text{dist}(i-1, t) + 1 & \text{delete} \\ \min_{b \in \{A,C,G,T\}} \text{dist}(i, b \bullet t[1..K-1]) + 1 & \text{insert} \end{cases}$$

Discuss the good, the bad, & the ugly of read error correction by spectra alignment



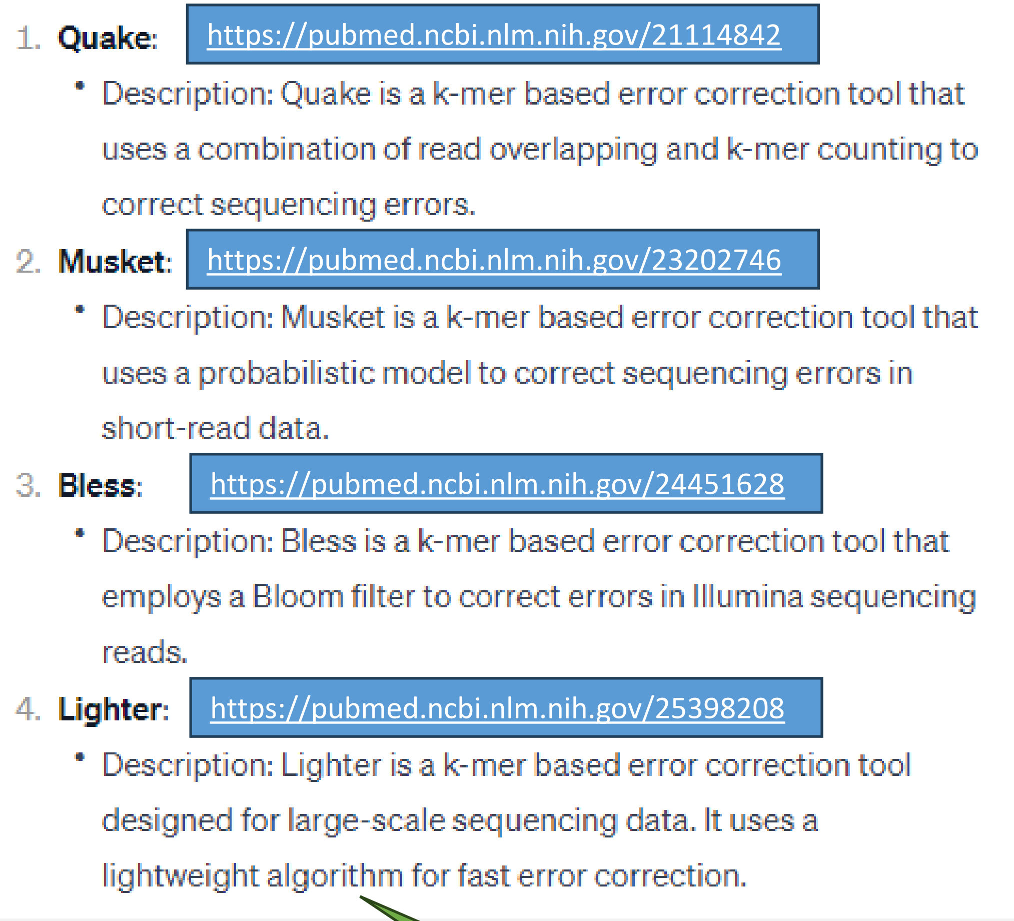
R[1..4] = AGTG

R[5] = C

R[6] = A

R[7] = G

Many modern & popular read error correction tools rely on K-mer counting & Bloom filter



1. **Quake:** https://pubmed.ncbi.nlm.nih.gov/21114842
   - Description: Quake is a k-mer based error correction tool that uses a combination of read overlapping and k-mer counting to correct sequencing errors.
2. **Musket:** https://pubmed.ncbi.nlm.nih.gov/23202746
   - Description: Musket is a k-mer based error correction tool that uses a probabilistic model to correct sequencing errors in short-read data.
3. **Bless:** https://pubmed.ncbi.nlm.nih.gov/24451628
   - Description: Bless is a k-mer based error correction tool that employs a Bloom filter to correct errors in Illumina sequencing reads.
4. **Lighter:** https://pubmed.ncbi.nlm.nih.gov/25398208
   - Description: Lighter is a k-mer based error correction tool designed for large-scale sequencing data. It uses a lightweight algorithm for fast error correction.

Check out Lighter especially. It does not do K-mer counting.

# of solid K-mers in human genome

n = size of Bloom filter
m = # of elements inserted
ε = false positive rate

~4.2 billion K-mers have freq = 1; assumed error K-mers

~2.8 billion K-mers have freq > 1; assumed solid K-mers

Optimal size of Bloom filter is $n = -2.08\ m\ (\ln \varepsilon)$ bits

$n = -2.08\ (2.8 \times 10^9)\ (\ln \varepsilon)$

$\approx 40 \times 10^9$ bits $\approx$ 5 GB at $\varepsilon = 0.01\%$

~420k error K-mers will test as solid K-mers

$\approx 54 \times 10^9$ bits $\approx$ 6.7 GB at $\varepsilon = 0.001\%$

~42k error K-mers will test as solid K-mers

Can use Bloom filter to keep solid K-mers for correcting read errors for human genome

A simple approach to Bloom filter-based read error correction

Keep solid K-mers in a Bloom filter H

For a read R, mark all positions $R[i .. i + K - 1]$ as solid if $R[i .. i + K - 1]$ is found in H

If a position $R[i]$ is not solid, replace $R[i]$ by $b \in \{A,C,G,T\}$ provided some of the following is found in H:

$b \bullet R[i + 1 .. i + K - 1]$

$R[i - K .. i - 1] \bullet b$

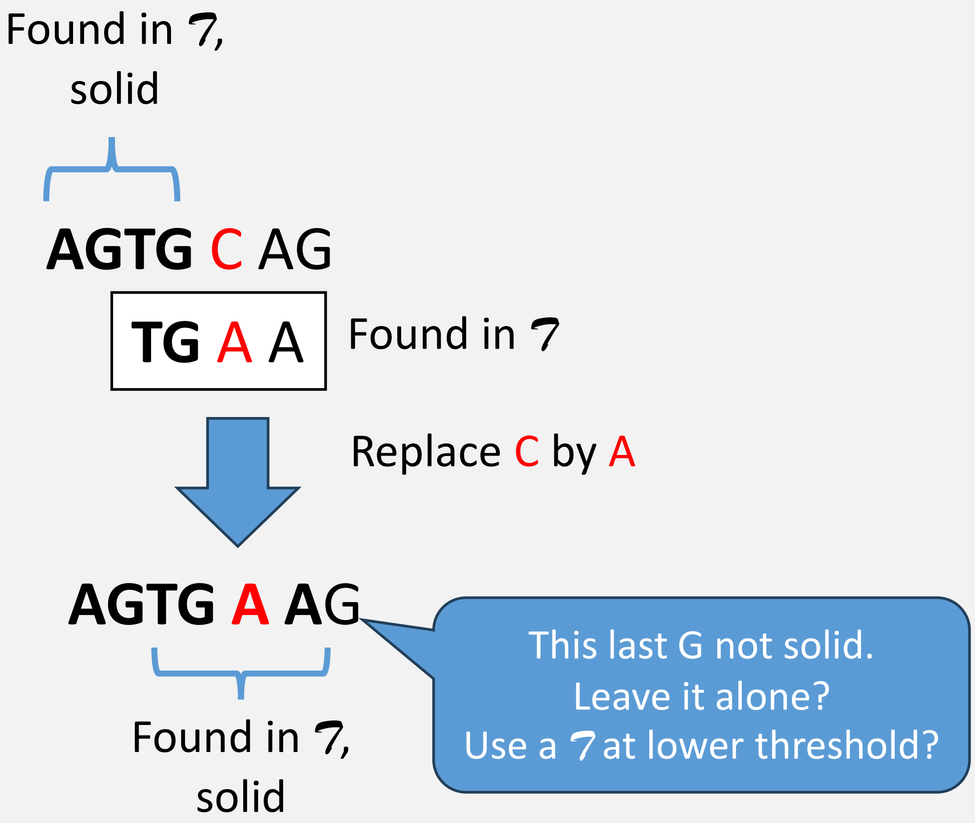
$R[i - j .. i - 1] \bullet b \bullet R[i + 1 .. i + K - j - 1]$, where $1 \leq j \leq K$

If more non-solid positions, repeat the last step

Example

R = AGTGCAG

$\mathcal{T}$ ={AAGT, **ACTT**,
AGTG, **CACT**,
TGAA, **TTCA** }



Found in $\mathcal{T}$,
solid

AGTG C AG

TG A A

Found in $\mathcal{T}$

Replace C by A

AGTG A AG

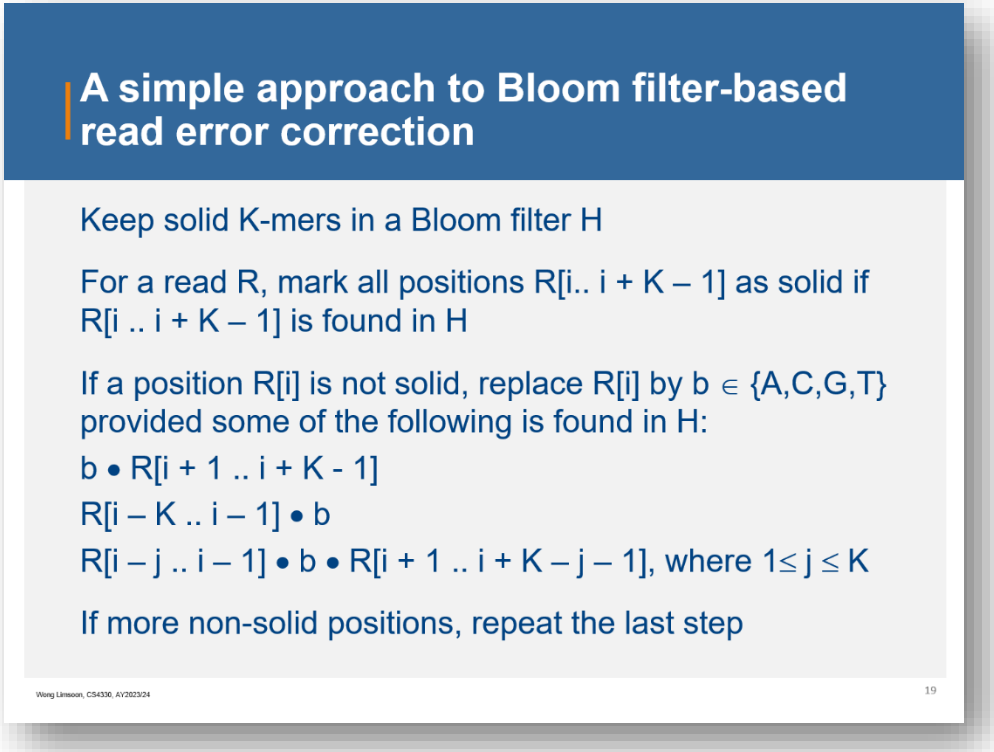
Found in $\mathcal{T}$,
solid

This last G not solid.
Leave it alone?
Use a $\mathcal{T}$ at lower threshold?

Exercise

Sometimes different "b" can be substituted, and hits found in H

How do you select the more likely one?



A simple approach to Bloom filter-based read error correction

Keep solid K-mers in a Bloom filter H

For a read R, mark all positions R[i.. i + K – 1] as solid if R[i .. i + K – 1] is found in H

If a position R[i] is not solid, replace R[i] by b $\in$ {A,C,G,T} provided some of the following is found in H:

b • R[i + 1 .. i + K - 1]

R[i – K .. i – 1] • b

R[i – j .. i – 1] • b • R[i + 1 .. i + K – j – 1], where $1 \leq j \leq K$

If more non-solid positions, repeat the last step

Wong Limsoon, CS4330, AY2023/24

19

Reminder: Low-frequency K-mers may not be errors

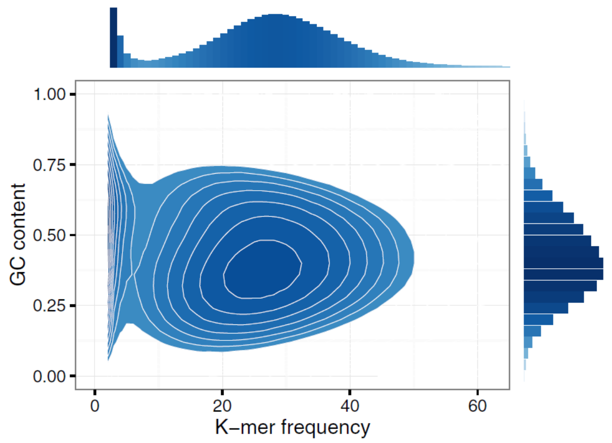


Fig. 3 A relation between *k*-mer frequency and GC-content. The bottom left panel shows the smoothed scatter plot between *k*-mer frequency and GC-content, the top left is the distribution of *k*-mer frequency, and the bottom right is the distribution of GC-content. It is clear that GC-content *k*-mers have relatively low frequency. The data shown in this example is obtained from the H. chromosome 14 with *k*-mer size of 25

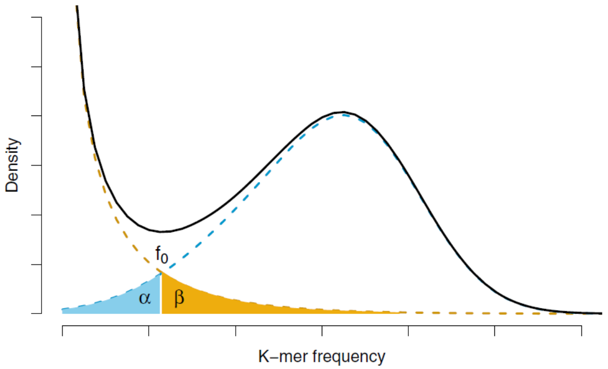


Fig. 1 Frequency distribution of both error-free and error-containing *k*-mers for a NGS data set. The frequency distribution of erroneous *k*-mers is represented by the dash orange line, while the distribution of the correct ones is shown as the dash sky-blue line. The solid black line is the distribution of all the k-mers. The α-labeled area is the proportion of correct *k*-mers having frequency less than f_0, while the β-labeled area is the proportion of erroneous *k*-mers having frequency greater than f_0

Zhao et al., Mining statistically-solid k-mers for accurate NGS error correction, *BMC Genomics* 19(S10):912, 2018

Problem

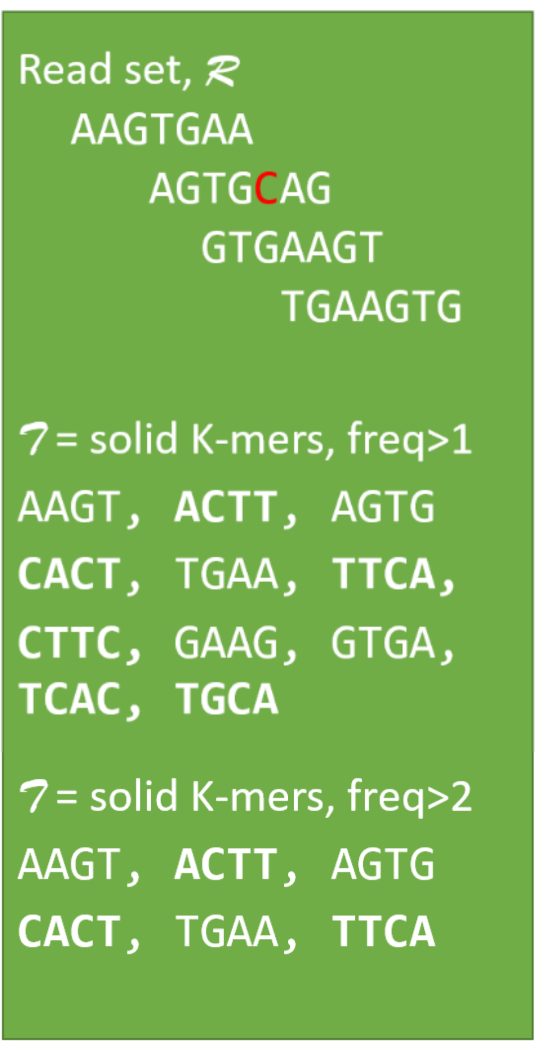
At freq > 1

Error K-mer TGCA $\in \mathcal{T}$

At freq > 2

Valid K-mer GAAG $\notin \mathcal{T}$

How to make $\mathcal{T}$ contain less error K-mers and more valid K-mers?



Read set, $\mathcal{R}$

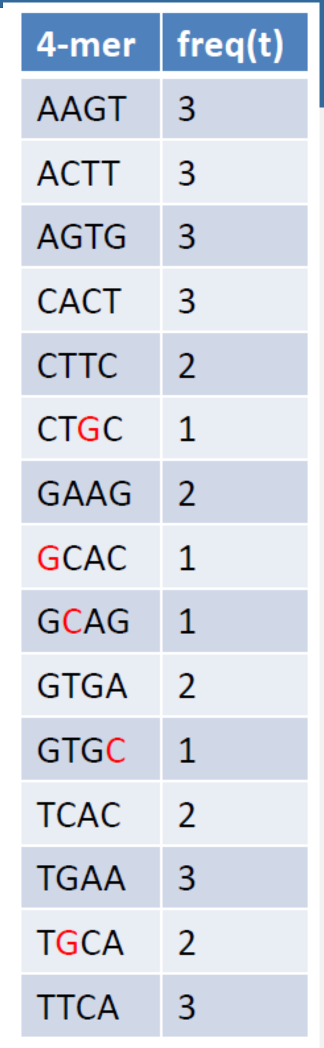
```
AAGTGAA
   AGTGCAG
      GTGAAGT
         TGAAGTG
```

$\mathcal{T}$ = solid K-mers, freq>1

AAGT, **ACTT**, AGTG **CACT**, TGAA, **TTCA**, **CTTC**, GAAG, GTGA, **TCAC**, **TGCA**

$\mathcal{T}$ = solid K-mers, freq>2

AAGT, **ACTT**, AGTG **CACT**, TGAA, **TTCA**



4-mer	freq(t)
AAGT	3
ACTT	3
AGTG	3
CACT	3
CTTC	2
CTGC	1
GAAG	2
GCAC	1
GCAG	1
GTGA	2
GTGC	1
TCAC	2
TGAA	3
TGCA	2
TTCA	3

ZEC approach

N(t) = { t' | t' is a K-mer in $\mathcal{R}$, Hamming(t,t') $\leq$ 1 }

C(t) = canonicalize N(t)

μ = mean of { freq(t') | t' $\in$ C(t) }

σ = std dev of { freq(t') | t' $\in$ C(t) }

Z(t) = [freq(t) - μ] / σ

τ > 0 is some threshold

Define t as solid if and only if freq(t) is large or Z(t) > τ

Zhao et al., *BMC Genomics* 19(S10):912, 2018

State of the art in read error correction performance

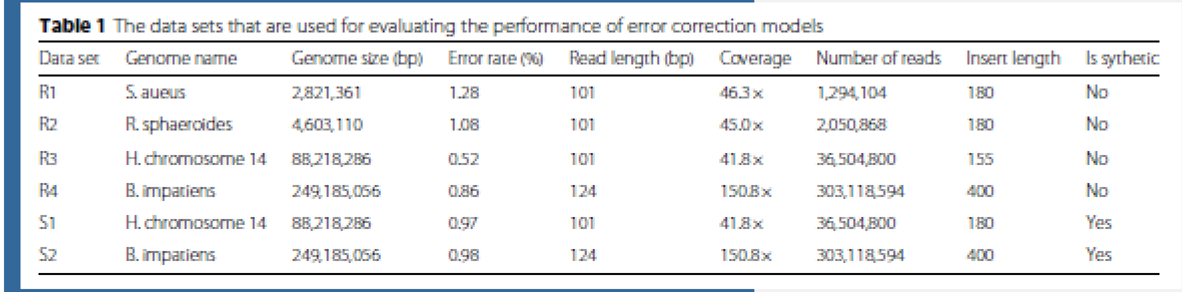
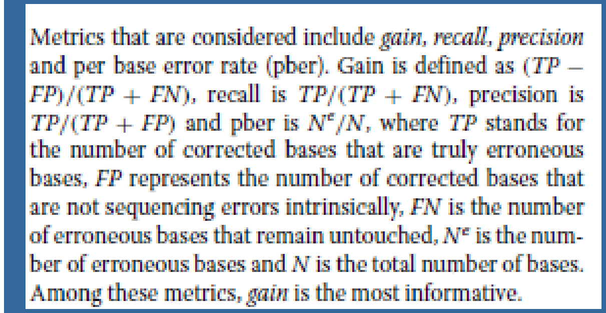
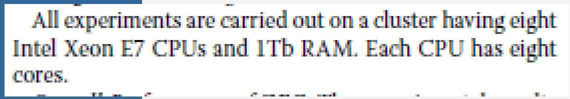


Table 1 The data sets that are used for evaluating the performance of error correction models

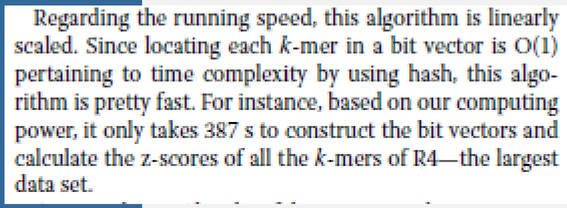
Data set	Genome name	Genome size (bp)	Error rate (%)	Read length (bp)	Coverage	Number of reads	Insert length	Is sythetic
R1	S. aueus	2,821,361	1.28	101	46.3 ×	1,294,104	180	No
R2	R. sphaeroides	4,603,110	1.08	101	45.0 ×	2,050,868	180	No
R3	H. chromosome 14	88,218,286	0.52	101	41.8 ×	36,504,800	155	No
R4	B. impatiens	249,185,056	0.86	124	150.8 ×	303,118,594	400	No
S1	H. chromosome 14	88,218,286	0.97	101	41.8 ×	36,504,800	180	Yes
S2	B. impatiens	249,185,056	0.98	124	150.8 ×	303,118,594	400	Yes



Metrics that are considered include *gain*, *recall*, *precision* and per base error rate (pber). Gain is defined as $(TP - FP)/(TP + FN)$, recall is $TP/(TP + FN)$, precision is $TP/(TP + FP)$ and pber is N^e/N, where TP stands for the number of corrected bases that are truly erroneous bases, FP represents the number of corrected bases that are not sequencing errors intrinsically, FN is the number of erroneous bases that remain untouched, N^e is the number of erroneous bases and N is the total number of bases. Among these metrics, *gain* is the most informative.



All experiments are carried out on a cluster having eight Intel Xeon E7 CPUs and 1Tb RAM. Each CPU has eight cores.



Regarding the running speed, this algorithm is linearly scaled. Since locating each k-mer in a bit vector is O(1) pertaining to time complexity by using hash, this algorithm is pretty fast. For instance, based on our computing power, it only takes 387 s to construct the bit vectors and calculate the z-scores of all the k-mers of R4—the largest data set.

Zhao et al., *BMC Genomics* 19(S10):912, 2018

Btw,
MEC is me ☺

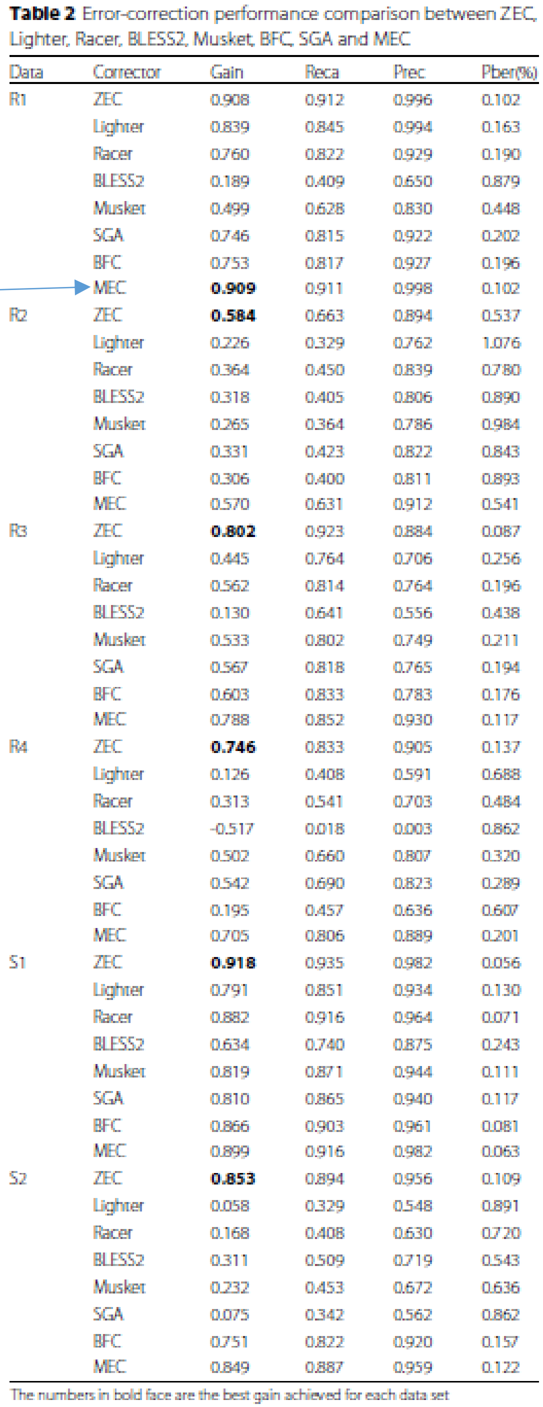


Table 2 Error-correction performance comparison between ZEC, Lighter, Racer, BLESS2, Musket, BFC, SGA and MEC

Data	Corrector	Gain	Reca	Prec	Pber(%)
R1	ZEC	0.908	0.912	0.996	0.102
	Lighter	0.839	0.845	0.994	0.163
	Racer	0.760	0.822	0.929	0.190
	BLESS2	0.189	0.409	0.650	0.879
	Musket	0.499	0.628	0.830	0.448
	SGA	0.746	0.815	0.922	0.202
	BFC	0.753	0.817	0.927	0.196
	MEC	**0.909**	0.911	0.998	0.102
R2	ZEC	**0.584**	0.663	0.894	0.537
	Lighter	0.226	0.329	0.762	1.076
	Racer	0.364	0.450	0.839	0.780
	BLESS2	0.318	0.405	0.806	0.890
	Musket	0.265	0.364	0.786	0.984
	SGA	0.331	0.423	0.822	0.843
	BFC	0.306	0.400	0.811	0.893
	MEC	0.570	0.631	0.912	0.541
R3	ZEC	**0.802**	0.923	0.884	0.087
	Lighter	0.445	0.764	0.706	0.256
	Racer	0.562	0.814	0.764	0.196
	BLESS2	0.130	0.641	0.556	0.438
	Musket	0.533	0.802	0.749	0.211
	SGA	0.567	0.818	0.765	0.194
	BFC	0.603	0.833	0.783	0.176
	MEC	0.788	0.852	0.930	0.117
R4	ZEC	**0.746**	0.833	0.905	0.137
	Lighter	0.126	0.408	0.591	0.688
	Racer	0.313	0.541	0.703	0.484
	BLESS2	-0.517	0.018	0.003	0.862
	Musket	0.502	0.660	0.807	0.320
	SGA	0.542	0.690	0.823	0.289
	BFC	0.195	0.457	0.636	0.607
	MEC	0.705	0.806	0.889	0.201
S1	ZEC	**0.918**	0.935	0.982	0.056
	Lighter	0.791	0.851	0.934	0.130
	Racer	0.882	0.916	0.964	0.071
	BLESS2	0.634	0.740	0.875	0.243
	Musket	0.819	0.871	0.944	0.111
	SGA	0.810	0.865	0.940	0.117
	BFC	0.866	0.903	0.961	0.081
	MEC	0.899	0.916	0.982	0.063
S2	ZEC	**0.853**	0.894	0.956	0.109
	Lighter	0.058	0.329	0.548	0.891
	Racer	0.168	0.408	0.630	0.720
	BLESS2	0.311	0.509	0.719	0.543
	Musket	0.232	0.453	0.672	0.636
	SGA	0.075	0.342	0.562	0.862
	BFC	0.751	0.822	0.920	0.157
	MEC	0.849	0.887	0.959	0.122

The numbers in bold face are the best gain achieved for each data set

Exercise

Why does ZEC work so well?

ZEC approach

$N(t) = \{ t' \mid t'$ is a K-mer in $\mathcal{R}$, Hamming$(t,t') \leq 1 \}$

$C(t) =$ canonicalize $N(t)$

$\mu =$ mean of $\{ freq(t') \mid t' \in C(t) \}$

$\sigma =$ std dev of $\{ freq(t') \mid t' \in C(t) \}$

$Z(t) = [freq(t) - \mu] / \sigma$

$\tau > 0$ is some threshold

Define t as solid if and only if freq(t) is large or $Z(t) > \tau$

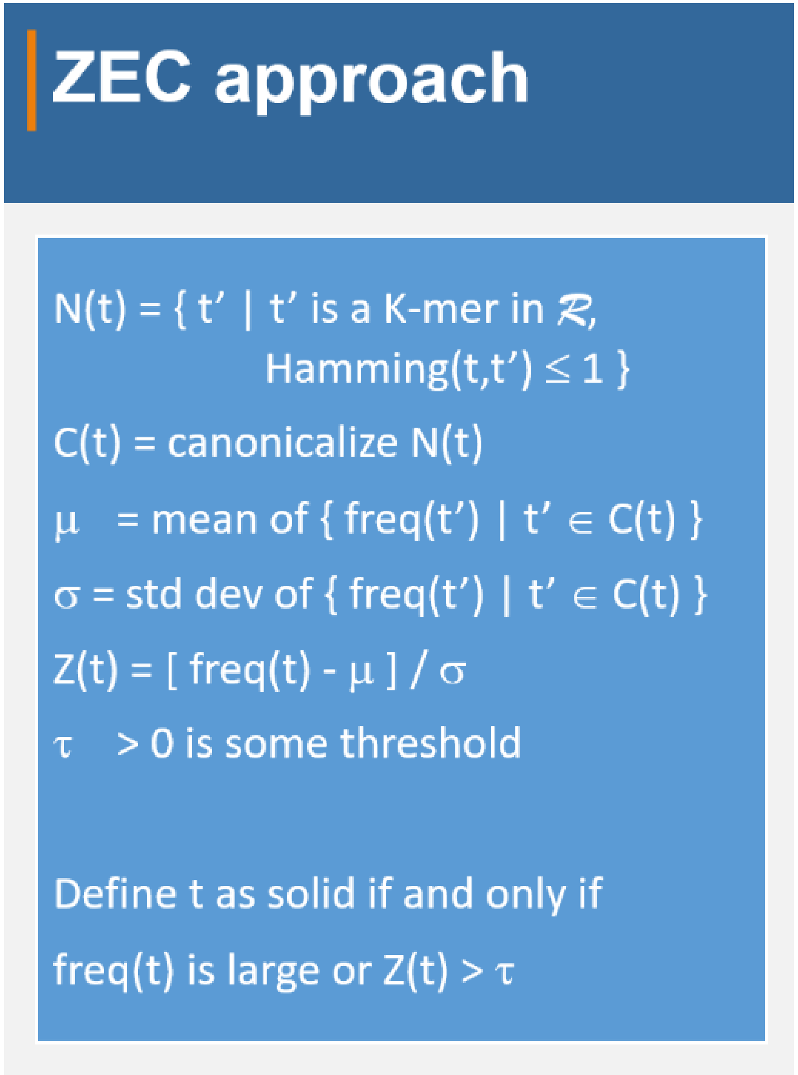
Exercise

ZEC requires computing μ and σ for each t in $\mathcal{R}$

Freq of every K-mer in $\mathcal{R}$ must be kept in memory

This requires terabytes of memory!

How does ZEC solve this problem?



ZEC approach

N(t) = { t' | t' is a K-mer in $\mathcal{R}$, Hamming(t,t') $\leq$ 1 }

C(t) = canonicalize N(t)

μ = mean of { freq(t') | t' $\in$ C(t) }

σ = std dev of { freq(t') | t' $\in$ C(t) }

Z(t) = [freq(t) - μ] / σ

τ > 0 is some threshold

Define t as solid if and only if

freq(t) is large or Z(t) > τ

Good to read

[Spectra alignment] M. Chaisson et al, “Fragment assembly with short reads”, Bioinformatics 20(13):2067-2074, 2004. https://pubmed.ncbi.nlm.nih.gov/15059830/

[ZEC] L. Zhao et al., “Mining statistically-solid k-mers for accurate NGS error correction”, BMC Genomics 19(S10):912, 2018. https://doi.org/10.1186/s12864-018-5272-y

[Lighter] L. Song et al., “Lighter: Fast and memory-efficient sequencing error correction without counting”, Genome Biology 15(11):509, 2014. https://pubmed.ncbi.nlm.nih.gov/25398208/

Good to read

[Musket] Y. Liu et al., “Musket: A multistage k-mer spectrum-based error corrector for Illumina sequence data”, Bioinformatics 29(3):308-315, 2013. https://pubmed.ncbi.nlm.nih.gov/23202746/

[MEC] L. Zhao et al., “MapReduce for accurate error correction of next-generation sequencing data”, Bioinformatics 33(23):3844-3851, 2017. https://pubmed.ncbi.nlm.nih.gov/28205674/