Read error correction using K-mers

Wong Limsoon

Acknowledgement: This set of slides were adapted from Ken Sung’s
Errors in reads greatly increase complexity of genome assembly

![De Bruijn graphs](image)

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 $freq(t) = \# \text{ of reads containing } t \text{ or its reverse complement}$.

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

![Example sequence](image)
Exercise

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

Reads with errors greatly increases complexity of genome assembly

We should discard these reads and not use them in genome assembly, no?
A K-mer \( t \) is said to be solid \( \mathcal{R} \) wrt a set of sequencing reads if \( \text{freq}(t) > M \), 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 \( \text{freq}(t) > M \)

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

AAGT, ACTT, AGTG
CACT, TGAA, TTCA

<table>
<thead>
<tr>
<th>4-mer</th>
<th>( \text{freq}(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAGT</td>
<td>3</td>
</tr>
<tr>
<td>ACTT</td>
<td>3</td>
</tr>
<tr>
<td>AGTG</td>
<td>3</td>
</tr>
<tr>
<td>CACT</td>
<td>3</td>
</tr>
<tr>
<td>CTTC</td>
<td>2</td>
</tr>
<tr>
<td>CTGC</td>
<td>1</td>
</tr>
<tr>
<td>GAAG</td>
<td>2</td>
</tr>
<tr>
<td>GCAC</td>
<td>1</td>
</tr>
<tr>
<td>GCAG</td>
<td>1</td>
</tr>
<tr>
<td>GTGA</td>
<td>2</td>
</tr>
<tr>
<td>GTGC</td>
<td>1</td>
</tr>
<tr>
<td>TCAC</td>
<td>2</td>
</tr>
<tr>
<td>TGAA</td>
<td>3</td>
</tr>
<tr>
<td>TGCA</td>
<td>2</td>
</tr>
<tr>
<td>TTCA</td>
<td>3</td>
</tr>
</tbody>
</table>
The read error correction problem

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

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

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

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, \( R \)
- AAGTGAA
- AGTG\( ^{\text{CAG}} \)
- GTGAAGT
- TGAAGTG

\( \mathcal{T} = \) solid K-mers, freq>1
- AAGT
- ACTT
- AGTG
- CACT
- TGAA
- TTCA
- CTTC
- GAAG
- GTGA
- TCAC
- TGCA

Which reads in \( R \) 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

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

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, $i = K$:

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

Recurrence:

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

Potential infinite loop!
The “dependency graph” is cyclic but non-negative

\[ R = AGTGCAG \]
\[ T = \{ AAGT, AGTG, GAAG, GTGA, TGAA, TGCA \} \]

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

\( R[1..4] = AGTG \)
\( R[5] = C \)
\( R[6] = A \)
\( R[7] = G \)

(mis)match = slant edge
delete = vertical edge
insert = horizontal edge
Spectra alignment via “shortest path” of dependency graph

Key lemma

\[ \text{dist}(i,t) = \text{length of shortest path from } v_s \text{ to } (i,t) \]

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

The dependency graph has \( O(|R| |T|) \) nodes and edges

\[ \therefore \text{Complexity of graph construction} = O(|R| |T|) \]

\[ \therefore \text{Complexity of shortest path finding} = O(|R| |T|) \]
Dijkstra’s shortest path algorithm

```python
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[]
```

Example

\[ R = \text{AGTGCAAG} \]
\[ T = \{ \text{AAGT}, \text{AGTG}, \text{GAAG}, \text{GTGA}, \text{TGAA}, \text{TGCA} \} \]

Min path length = 1
Corrected read = \text{AGTGAAG}

Recurrence:
\[
dist(i, t) = \min_{b \in \{A, C, G, T\}} \begin{cases} 
\text{match} & \text{dist}(i - 1, b \cdot t[1..K-1]) + p(R[i], t[K]) \\
\text{delete} & \text{dist}(i - 1, t) + 1 \\
\text{insert} & \min_{b \in \{A, C, G, T\}} \text{dist}(i, b \cdot t[1..K-1]) + 1 
\end{cases}
\]
Exercise

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

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

- \( R[1..4] = AGTG \)
- \( R[5] = C \)
- \( R[6] = A \)
- \( R[7] = G \)
~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

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

\[ n \approx 40 \times 10^9 \text{ bits} \approx 5 \text{ GB at } \varepsilon = 0.01\% \]
\[ n \approx 54 \times 10^9 \text{ bits} \approx 6.7 \text{ GB at } \varepsilon = 0.001\% \]

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

\[ n = \text{size of Bloom filter} \]
\[ m = \text{# of elements inserted} \]
\[ \varepsilon = \text{false positive rate} \]
Many modern & popular read error correction tools rely on K-mer counting & Bloom filter

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quake</td>
<td>Quake is a k-mer based error correction tool that uses a combination of read overlapping and k-mer counting to correct sequencing errors.</td>
<td><a href="https://pubmed.ncbi.nlm.nih.gov/21114842">https://pubmed.ncbi.nlm.nih.gov/21114842</a></td>
</tr>
<tr>
<td>Musket</td>
<td>Musket is a k-mer based error correction tool that uses a probabilistic model to correct sequencing errors in short-read data.</td>
<td><a href="https://pubmed.ncbi.nlm.nih.gov/23202746">https://pubmed.ncbi.nlm.nih.gov/23202746</a></td>
</tr>
<tr>
<td>Bless</td>
<td>Bless is a k-mer based error correction tool that employs a Bloom filter to correct errors in Illumina sequencing reads.</td>
<td><a href="https://pubmed.ncbi.nlm.nih.gov/24451628">https://pubmed.ncbi.nlm.nih.gov/24451628</a></td>
</tr>
<tr>
<td>Lighter</td>
<td>Lighter is a k-mer based error correction tool designed for large-scale sequencing data. It uses a lightweight algorithm for fast error correction.</td>
<td><a href="https://pubmed.ncbi.nlm.nih.gov/25398208">https://pubmed.ncbi.nlm.nih.gov/25398208</a></td>
</tr>
</tbody>
</table>

Check out Lighter especially. It does not do K-mer counting.
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 \( H \):
\[
\begin{align*}
&b \bullet R[i + 1 .. i + K - 1] \\
&R[i - K .. i - 1] \bullet b \\
&R[i - j .. i - 1] \bullet b \bullet R[i + 1 .. i + K - j - 1], \text{ where } 1 \leq j \leq K
\end{align*}
\]

If more non-solid positions, repeat the last step
Example

\[ R = \text{AGTGCAG} \]

\[ \mathcal{F} = \{ \text{AAGT, ACTT, AGTG, CACT, TGAA, TTCA} \} \]

Found in \( \mathcal{F} \), solid

\[ \text{AGTG C AG} \]

Found in \( \mathcal{F} \)

Replace C by A

\[ \text{AGTG A AG} \]

Found in \( \mathcal{F} \), solid

This last G not solid.
Leave it alone?
Use a \( \mathcal{F} \) 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 ∈ {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 ≤ j ≤ K

If more non-solid positions, repeat the last step
Reminder:
Low-frequency K-mers may not be errors

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

At freq > 1
Error K-mer TGCA ∈ 7

At freq > 2
Valid K-mer GAAG ∉ 7

How to make 7 contain less error K-mers and include more valid K-mers?

Read set, $\mathcal{R}$
- AAGT
- TGCA
- AGTG
- CACT
- GTGA
- AGGT
- TGAAGT

7 = solid K-mers, freq>1
- AAGT
- ACTT
- AGTG
- CACT
- TGAA
- TTCA

7 = solid K-mers, freq>2
- AAGT
- ACTT
- AGTG
- CACT
- TGAA
- TTCA

<table>
<thead>
<tr>
<th>4-mer</th>
<th>freq(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAGT</td>
<td>3</td>
</tr>
<tr>
<td>ACTT</td>
<td>3</td>
</tr>
<tr>
<td>AGTG</td>
<td>3</td>
</tr>
<tr>
<td>CACT</td>
<td>3</td>
</tr>
<tr>
<td>CTTC</td>
<td>2</td>
</tr>
<tr>
<td>CTGC</td>
<td>1</td>
</tr>
<tr>
<td>GAAG</td>
<td>2</td>
</tr>
<tr>
<td>GCAC</td>
<td>1</td>
</tr>
<tr>
<td>GCAG</td>
<td>1</td>
</tr>
<tr>
<td>GTGA</td>
<td>2</td>
</tr>
<tr>
<td>GTGC</td>
<td>1</td>
</tr>
<tr>
<td>TCAC</td>
<td>2</td>
</tr>
<tr>
<td>TGAA</td>
<td>3</td>
</tr>
<tr>
<td>TGCA</td>
<td>2</td>
</tr>
<tr>
<td>TTCA</td>
<td>3</td>
</tr>
</tbody>
</table>
State of the art in read error correction, ZEC

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

Btw, MEC is me 😊
Must read

Spectra alignment


ZEC


Lighter

Good to read

Musket


MEC