Alignment-free genomics
# Sequence Search

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567890123</td>
<td>1234567890123</td>
</tr>
<tr>
<td>T: xabxyabxyabxz</td>
<td>T: xabxyabxyabxz</td>
</tr>
<tr>
<td>P: abxyabxz</td>
<td>P: abxyabxz</td>
</tr>
<tr>
<td>abxyabxz</td>
<td>abxyabxz</td>
</tr>
<tr>
<td>^^^^^^^*</td>
<td>^^^^^^^*</td>
</tr>
<tr>
<td>abxyabxz</td>
<td>abxyabxz</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>abxyabxz</td>
<td>abxyabxz</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>abxyabxz</td>
<td>abxyabxz</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>abxyabxz</td>
<td>abxyabxz</td>
</tr>
</tbody>
</table>


Image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 1.1
Sequence Search

Given a pattern $p$ and a text $q$, find $p$ in $q$

- naive solution is $O(mn)$ time, $m = |p|$, $n = |q|$
- improved to $O(n+m)$ by Boyer and More\(^1\)
- later $O(n)$ by Knuth, Morris and Pratt\(^2\)

---


Image: Gusfield, D. Algorithms on Strings, Trees and Sequences, 1997. Figure 1.1
Sequence Search

Given a pattern $p$ and a text $q$, find $p$ in $q$

- naive solution is $O(mn)$ time, $m = |p|$, $n = |q|$
- improved to $O(n+m)$ by Boyer and More$^1$
- later $O(n)$ by Knuth, Morris and Pratt$^2$

What if $n$ is very large?

---


image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 1.1
Suffix Trie/Tree

\[ P = \text{abx} \]
\[ Q = \text{xabxaxc} \]

123456

Image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 5.1 (modified)
Suffix Trie/Tree

Let $T$ be a rooted tree
• where each edge is labeled by a distinct character $a \in \Sigma$, and
• each leaf $l$ labels a suffix of $\sigma$ such concatenating the labels of the edges from the root to $p$ form the suffix.
• Finding $p$ in $T$ takes $O(m)$ time.
• Finding all instances is also faster.
• Requires $\Theta(n|\Sigma|)$ space!
Suffix Trie/Tree

Let $T$ be a rooted tree

• where each edge is labeled by a distinct character $a \in \Sigma$, and
• each leaf labels a suffix of $q$ such concatenating the labels of the edges from the root to $p$ form the suffix.
• Finding $p$ in $T$ takes $O(m)$ -time.
• Finding all instances is also faster.
• Requires $\Theta(n|\Sigma|)$ space!

But, what if $n = 3,000,000,000$?

image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 5.1 (modified)
Suffix Arrays

Store two arrays
• \( \text{pos}(i) \) — which are the start position of suffixes in lexicographic order, and
• \( \text{lcp}(i, j) \) — which stores the longest common prefix between positions \( i \) and \( j \).
• Takes \( O(n) \) space.
• Search can be conducted in \( O(m + \log n) \)-time.
Burrows-Wheeler Transform

mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
mississippi
Burrows-Wheeler Transform

Store the last column of the rotated sorted suffix list
• Can be easily compressed because of the repetitiveness
• When used along with the genomic sequence can quickly recover the original sequence
• Ferrangina and Manzini later made advances for faster search
Burrows-Wheeler Transform

Store the last column of the rotated sorted suffix list
- Can be easily compressed because of the repetitiveness
- When used along with the genomic sequence can quickly recover the original sequence
- Ferrangina and Manzini later made advances for faster search

What if we want to find positions with some changes?
Alignment

Given
- two sequences \( p \) and \( q \) over an alphabet \( \Sigma \), and
- an alignment objective function.

Find an \( m \times 2 \) matrix ( \( m > \max(|p|,|q|) \) )
- where each row represents one of the sequences with inserted gap characters (‘-’ \( \not\in \Sigma \)), and
- is optimal under the objective function.

\[
p = \text{GATTACA} \quad \rightarrow \quad \text{G-ATTACA} \\
q = \text{GCATGCT} \quad \rightarrow \quad \text{GCA-TGCT}
\]
Alignment

Can be solved in
• $O(|p| \cdot |q|)$ time using Needleman–Wunsch algorithm\(^1\)
• Extended to local alignment by Smith and Waterman\(^2\)

With local alignment, easily find the best location of a small string within another even if there are errors.


Seed and Extend

Given a pattern \( p \) and a text \( q \), find \( p \) in \( q \)
- select a substring \( p' \) from \( p \)
- search for \( p' \) in \( q \) using an exact search method
- only perform alignment on a small region around locations of \( p' \)
- Requiring multiple seeds can further reduce search locations and/or increase the number of errors allowed
Quasi-alignment

Given a pattern $p$ and a text $q$, find $p$ in $q$
- by first finding the set $P$ of all overlapping subsequences of length $k$
- find the locations of $p'$ in $P$ in $q$
- if there is a region where a large percentage of $P$ are found very close call that location a location of $p$. 
k-mer Counting

For a given sequence $q$ and value $k$
- determine the list of unique $k$-length strings in $q$, and
- count the frequency of each.
- Can be used to quickly compare two sequences.
- Problems arise in keeping the hash table (naively $\Sigma^k$ entries)

$q = \text{xabxyabxyabxz}$
$k = 3$

<table>
<thead>
<tr>
<th>String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>abx</td>
<td>3</td>
</tr>
<tr>
<td>bxy</td>
<td>2</td>
</tr>
<tr>
<td>bxz</td>
<td>1</td>
</tr>
<tr>
<td>xab</td>
<td>1</td>
</tr>
<tr>
<td>xya</td>
<td>2</td>
</tr>
<tr>
<td>yab</td>
<td>2</td>
</tr>
</tbody>
</table>
Minimizer schemes

Given a string $q$, and values $k$ and $w$

- for each substring of $w$ $k$-mers
- only select the minimum.
- This reduces the total number of $k$-mers that must be considered.
- Changing the ordering can impact the number of unique $k$-mers.

$q = xabxyabxyabxz$
$k = 3$
$w = 2$

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>minimizer count</th>
</tr>
</thead>
<tbody>
<tr>
<td>abx</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>bxy</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>bxz</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>xab</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>xya</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>yab</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
**Metagenomics**

**Given a group of sequences $Q$**
- group $q$ in $Q$ so similar sequences are from the same (possibly unknown) organism.
- Similarity can be measured using edit distance (alignment), $k$-mer counts, etc.
Naive sequence search

\[ P = \text{abx} \]
\[ Q = \text{xabxac} \]

Suffix trees

```
P = abx
Q = xabxac
```

Suffix arrays

```
<table>
<thead>
<tr>
<th>pos</th>
<th>lcp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
</tbody>
</table>
```

BWT

```
<table>
<thead>
<tr>
<th>pos</th>
<th>lcp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>$mississippi</td>
</tr>
<tr>
<td>$</td>
<td>mississippi</td>
</tr>
<tr>
<td>$</td>
<td>pippismississippi</td>
</tr>
<tr>
<td>$</td>
<td>mississippi</td>
</tr>
<tr>
<td>$</td>
<td>mississippi</td>
</tr>
<tr>
<td>$</td>
<td>mississippi</td>
</tr>
<tr>
<td>$</td>
<td>mississippi</td>
</tr>
</tbody>
</table>
```

Alignment

Needleman-Wunsch

```
match = 1  mismatch = -1  gap = -1
```

Quasi-alignment

```
p
q = xabxyabxyabxz
k = 3
```

k-mer counting

```
<table>
<thead>
<tr>
<th>q = xabxyabxyabxz</th>
<th>k = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>abx</td>
<td>3</td>
</tr>
<tr>
<td>bxy</td>
<td>2</td>
</tr>
<tr>
<td>bxz</td>
<td>2</td>
</tr>
<tr>
<td>xab</td>
<td>1</td>
</tr>
<tr>
<td>xya</td>
<td>1</td>
</tr>
<tr>
<td>yab</td>
<td>2</td>
</tr>
</tbody>
</table>
```
Minimizers

Given a set of sequences, compute all of the pairwise overlaps
Minimizers

Given a set of sequences, compute all of the pairwise overlaps
Minimizers

Given a set of sequences, compute all of the pairwise overlaps

$O(mn)$ time for each pair!
Minimizers

Given a set of sequences, compute all of the pairwise overlaps
Minimizers

Given a set of sequences, compute all of the pairwise overlaps
Minimizers

Given a set of sequences, compute all of the pairwise overlaps

When sorted exact matches appear together and can be mapped easily

Slide adapted from those of Guillaume Marçais
Minimizers

Given a set of sequences, compute all of the pairwise overlaps
Minimizers

Given a set of sequences, compute all of the pairwise overlaps
Minimizers

Given a set of sequences, compute all of the pairwise overlaps

Slide adapted from those of Guillaume Marçais
Minimizers

How should you choose the k-mers from a string

- Small number of k-mers
- Cover the whole string
- Long overlaps should have large numbers of matched k-mers

Store exactly every $w^{th}$ mer

Storage and comparisons reduced by factor of $1/w$

Slide adapted from those of Guillaume Marçais
Minimizers

How should you choose the k-mers from a string

- Small number of k-mers
- Cover the whole string
- Long overlaps should have large numbers of matched k-mers

Store exactly every $w^{\text{th}}$ mer

Storage and comparisons reduced by factor of $1/w$

Large overlap, no matched k-mers

Slide adapted from those of Guillaume Marçais
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.

\[
\begin{array}{c}
111 \\
110 \\
101 \\
100 \\
011 \\
010 \\
001 \\
000 \\
\end{array}
\]

$S = 10001$

Slide adapted from those of Guillaume Marçais
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.

Slide adapted from those of Guillaume Marçais
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.

Slide adapted from those of Guillaume Marçais
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers

- choose the smallest $k$-mer as the fingerprint.
Minimizers

How should you choose the $k$-mers from a string

- Small number of $k$-mers
- Cover the whole string
- Long overlaps should have large numbers of matched $k$-mers

For each window of $w$ $k$-mers
- choose the smallest $k$-mer as the fingerprint.

Density of an order $o$ of $(o, S, k)$:

$$d(o, S, k) = \frac{\# \text{ of selected positions}}{|S| - k + 1}$$
Minimizers

Lexicographic order: every $k$-mer picked at least once
Minimizers

Lexicographic order: every $k$-mer picked at least once

Optimized order: top 3 $k$-mers never picked

Slide adapted from those of Guillaume Marçais
Universal hitting set

A universal set $\mathcal{U}_{k,w}$ is

- a set of $k$-mers such that
- all string of length $w+k-1$ contains one $k$-mers from the set
- that minimizes the size of the set.
- Found using the de Bruijn graph of $k$-mers by first selecting nodes that intersect all cycles (decycling)
- then additional nodes to intersect long paths.
Universal hitting set

A universal set $U_{k,w}$ is
- a set of $k$-mers such that
- all string of length $w+k-1$ contains one $k$-mers from the set
- that minimizes the size of the set.
- Found using the de Bruijn graph of $k$-mers by first selecting nodes that
  intersect all cycles (decycling)
- then additional nodes to intersect long paths.

Universal set ordering
- Given a universal set $U_{k,w}$
- rank all $k$-mers from the set lower than any other.
Universal hitting set

$k = 8$, size of universal hitting set compared to decycling set

Slide adapted from those of Guillaume Marçais
Minimizers

Density for human chromosome 19, $k=7$, $w=11$
Lempel-Ziv Complexity

Let the distance between strings be:

- compressibility of the concatenated string compared to them individually compressed.
- Similar strings will have a higher compression ratio.

**Example:***

<table>
<thead>
<tr>
<th>Query sequences</th>
<th>Lempel-Ziv complexity</th>
<th>Normalized compression distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>x ATGTGTG</td>
<td>y CATGTG</td>
<td>xy ATGTGTGCATGTG</td>
</tr>
<tr>
<td>c(x)=4</td>
<td>c(y)=5</td>
<td>c(xy)=7</td>
</tr>
</tbody>
</table>

\[
\frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} = \frac{7-4}{5} = 0.6
\]

Kullback-Leibler distance

Use the difference in information content to determine the distance between strings.

<table>
<thead>
<tr>
<th>x</th>
<th>ATGTGTG</th>
<th>y</th>
<th>CATGTG</th>
<th>query sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1^x )</td>
<td>ATG</td>
<td>( w_1^y )</td>
<td>CATG</td>
<td>word size: 1</td>
</tr>
<tr>
<td>( w_1 = w_1^x \cup w_1^y )</td>
<td>A C G T</td>
<td>union</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_i^x )</td>
<td>1 0 3 3</td>
<td>( c_i^y )</td>
<td>1 1 2 2</td>
<td>word counts</td>
</tr>
<tr>
<td>( p_i^x )</td>
<td>0.14 0.00 0.41</td>
<td>( p_i^y )</td>
<td>0.17 0.17 0.31 0.31</td>
<td>word frequencies</td>
</tr>
<tr>
<td>[ \sum_{i=1} \log \left( \frac{p_i^x}{p_i^y} \right) ]</td>
<td>0.14 \log \left( \frac{0.14}{0.17} \right) + 0 + 0.43 \log \left( \frac{0.43}{0.33} \right) + 0.43 \log \left( \frac{0.43}{0.33} \right) = 0.24</td>
<td>Kullback-Leibler divergence</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Metagenomics

- Normally one experiment -> one organism
- Multiple organisms makes for a harder problem
- Not knowing what possible species are there makes it harder still
Metagenomics

Given a set of sequences \( Q \)
- group similar sequences together.
- Equivalent to the classical problem of clustering.
- Distance metrics become important.
Metagenomics

Given a set of sequences, compute all of the pairwise distance
Metagenomics

Once the data is clustered
- identify certain groups
- assemble clusters
- generate phylogeny
- ....

image source: Laczny, et al. Alignment-free Visualization of Metagenomic Data by Nonlinear Dimension Reduction Scientific Reports, 2014
### Table 1: Alignment-free sequence comparison tools available for next-generation sequencing data analysis

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Platform(s)</th>
<th>Website(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAF</td>
<td>Phylogeny reconstruction directly from unassembled raw sequence data from whole-genome sequencing projects, provides bootstrap support to assess uncertainty in the tree topology (k-mer based)</td>
<td>Software (Python)</td>
<td><a href="https://sourceforge.net/projects/trowel-ec/">SourceForge</a>; <a href="https://github.com/fanhuan/AAF">GitHub</a></td>
</tr>
<tr>
<td>KSNP.v3</td>
<td>Reference-free SNP identification and estimation of phylogenetic trees using SNPs (based on k-mer analysis)</td>
<td>Software (C)</td>
<td><a href="https://sourceforge.net/projects/kSNPv3/">SourceForge</a>; <a href="https://github.com/fanhuan/AAF">GitHub</a></td>
</tr>
<tr>
<td>NGS-MC</td>
<td>Phylogeny of species based on NGS reads using alignment-free sequence dissimilarity measures d*, and d2 under different Markov chain models (using k-mers)</td>
<td>R package</td>
<td><a href="http://www.ri.cuc.edu.tw/~funProject/NGS-MC/NGS-MC.html">SourceForge</a></td>
</tr>
<tr>
<td>CLARK</td>
<td>Taxonomic classification of metagenomic reads to known bacterial genomes using k-mer search and LCA assignment</td>
<td>Software (C++)</td>
<td><a href="http://clark.cs.ucr.edu">GitHub</a></td>
</tr>
<tr>
<td>FOCUS</td>
<td>Reports organisms present in metagenomic samples and profiles their abundances (uses composition-based approach and non-negative least squares for prediction)</td>
<td>Web service</td>
<td><a href="http://edwards.sdsu.edu/FOCUS/">GitHub</a></td>
</tr>
<tr>
<td>CLARK</td>
<td>Taxonomic classification of metagenomic reads to known bacterial genomes using k-mer search and LCA assignment</td>
<td>Web service</td>
<td><a href="http://clark.cs.ucr.edu">GitHub</a></td>
</tr>
<tr>
<td>GMM</td>
<td>Estimation of abundances of microbial genomes in metagenomic samples (k-mer based)</td>
<td>Software (Java)</td>
<td><a href="https://github.com/jsh/wkGMM">GitHub</a></td>
</tr>
<tr>
<td>MinV</td>
<td>Species identification using assembled or unassembled Illumina, Pacifica, and ONT data (based on k-mer to dimensional-reduction technique)</td>
<td>Software (C++)</td>
<td><a href="https://github.com/mdb/mv">GitHub</a></td>
</tr>
<tr>
<td>Kraken</td>
<td>Taxonomic assignment in metagenome analysis by exact k-mer search, LCA assignment of short reads based on a comprehensive sequence database</td>
<td>Software (C++)</td>
<td><a href="https://github.com/kb/-software/kraken">GitHub</a></td>
</tr>
<tr>
<td>LMMAT</td>
<td>Assignment of taxonomic labels to reads by k-mers searches in precomputed database</td>
<td>Software (C++/Python)</td>
<td><a href="https://sourceforge.net/projects/lmmat/">SourceForge</a></td>
</tr>
<tr>
<td>stringMLST</td>
<td>k-mer-based tool for MLST directly from the genome sequencing reads</td>
<td>Software (Python)</td>
<td><a href="https://jordan.biology.gatech.edu/page/software/stringMLST">GitHub</a></td>
</tr>
<tr>
<td>Taxonamer</td>
<td>k-mer-based ultradaf metagenomics tool for assigning taxonomy to sequencing reads from clinical and environmental samples</td>
<td>Web service</td>
<td><a href="https://taxonomer.scribio.io">GitHub</a></td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>iEJ-tool</strong></td>
<td>Word-based (k-tuple) comparison (pairwise dissimilarity matrix using ASG measure) of metatranscriptomic samples from NGS reads</td>
<td>Software (Python/R)</td>
<td><a href="https://code.google.com/p/polychord-tools/">GitHub</a></td>
</tr>
<tr>
<td><strong>VirHostMatcher</strong></td>
<td>Prediction of hosts from metagenomic viral sequences based on ONF using various distance measures (e.g., d2)</td>
<td>Software (C++)</td>
<td><a href="https://github.com/jessieren/VirHostMatcher">GitHub</a></td>
</tr>
<tr>
<td><strong>MetaFast</strong></td>
<td>Statistics calculation of metagenomic sequences and the distances between them based on assembly using de Bruijn graphs and Bray-Curtis dissimilarity measure</td>
<td>Software (Java)</td>
<td><a href="https://github.com/citolab/metafast">GitHub</a></td>
</tr>
</tbody>
</table>

Sailfish

02-715
6 February 2018
Why?

Quantification is used for
  • differential expression
  • disease sub-typing, and
  • cancer progression analysis
Alignment-free

Alignment is slow

The bar chart shows the time taken for alignment and quantification for different tools on two datasets: SRX016366 and Synthetic. The tools compared are Sailfish, RSEM, eXpress, and Cufflinks. The y-axis represents time in hours, and the x-axis represents the tools. The chart indicates that alignment is significantly slower than quantification for all tools, with Sailfish taking 6.03 hours and RSEM taking 10.19 hours on SRX016366, and 18.55 hours and 6.96 hours on the Synthetic dataset, respectively.
Alignment-free

Alignment is slow

Why else?
Alignment-free

Alignment is slow

Why else?
• Error tolerance
Alignment-free

Alignment is slow

Why else?
• Error tolerance
• Storage size
  - Sailfish on human = 3.1G
  - Bowtie alignment = 15.5G
Alignment-free

Alignment is slow

Why else?
  • Error tolerance
  • Storage size
    - Sailfish on human = 3.1G
    - Bowtie alignment = 15.5G
  • Memory usage
The Index

The index is the key it contains
• a perfect hash of k-mers (in T) to indices
• a count of k-mer use
• maps from T to k-mers and vice versa
Equivalence Classes

An equivalence class is

• all k-mers
• that appear in the same set of transcripts
• with the same frequency

For k=20 and the human experiment

• number of k-mers: $4^{20} = 2^{40} =$ 1,099,511,627,776
• number of k-mers in the human transcriptome = 60,504,111
• number of k-mers in the human read set = 39,393,132
• number of distinct equivalence classes = 151,385
  (0.38%)
Why is $l_i' = l - k + 1$?
Bias Correction

Expression prediction can be impacted by
• fragment length
• CG content
• dinucleotide frequency

Corrections are made following procedures in Zeng, et al. (2011)
• measure length, GC and DN frequency
• find PCs for GC and DN
• fit a correction function
Real Data

\[ \text{TPM}_i = 10^6 \mu'_i \]

\[ \text{KPKM}_i = \frac{C_i}{N \cdot 10^3} = \frac{10^9 C_i}{l_i N} \approx \frac{10^9 \mu_i}{N} \]

k-mers per kilobase, per million mapped k-mers

<table>
<thead>
<tr>
<th>Human brain tissue</th>
<th>Sailfish</th>
<th>RSEM</th>
<th>eXpress</th>
<th>Cufflinks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pearson</td>
<td>0.85</td>
<td>0.82</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>Spearman</td>
<td>0.84</td>
<td>0.80</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Simulation

\[ \text{PE}(x_i, y_i) = 100 \times \frac{|x_i - y_i|}{x_i} \]

\[ \text{RMSE}(x, y) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \]

<table>
<thead>
<tr>
<th></th>
<th>Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sailfish</td>
<td>0.96</td>
</tr>
<tr>
<td>RSEM</td>
<td>0.76</td>
</tr>
<tr>
<td>eXpress</td>
<td>0.96</td>
</tr>
<tr>
<td>Cufflinks</td>
<td>0.95</td>
</tr>
<tr>
<td>Pearson</td>
<td>0.77</td>
</tr>
<tr>
<td>Spearman</td>
<td>0.77</td>
</tr>
<tr>
<td>RMSE</td>
<td>6.06</td>
</tr>
<tr>
<td>medPE</td>
<td>6.50</td>
</tr>
<tr>
<td></td>
<td>10.05</td>
</tr>
<tr>
<td></td>
<td>12.48</td>
</tr>
<tr>
<td></td>
<td>14.06</td>
</tr>
<tr>
<td></td>
<td>12.42</td>
</tr>
</tbody>
</table>
Take Aways

What was new
  • alignment-free quantification is fast
  • similar accuracy to alignment-based methods
  • EM efficiently takes care of multi-mapped reads
  • single input parameter

Problems
  • biases are not addressed fully
Going beyond k-mer counts

Salmon


Advanced Bias Correction

Salmon extends the BC from Sailfish
• considers 5’- and 3’- sequencing bias
• fragment-level GC bias
• length-bias

\[
\ell'_i = \sum_{j=1}^{\ell_i} \sum_{k=1}^{f_{i}(j,L)} \frac{b_{gc}^+(t_i, j, j+k) \cdot b_{5'}^+(t_i, j) \cdot b_{3'}^+(t_i, j+k)}{b_{gc}^-(t_i, j, j+k) \cdot b_{5'}^-(t_i, j) \cdot b_{3'}^-(t_i, j+k)} \cdot \Pr\{X = j\}
\]
Accuracy of new methods

![Bar chart and box plot showing accuracy measures for different methods.](image-url)
Suggested Reading

Metagenomics

Suggested Reading

Quantification


- D. C. Wu, J. Yao, K. S. Ho, A. M. Lambowitz, C. O. Wilke, "Limitation of alignment-free tools in total RNA-seq quantification", bioRxiv, 10.1101/246967 (review)

Minimizers


Suggested Reading

**Storage/Search**
Suggested Reading

**k-mer Counting**

Suggested Reading

**Phylogeny**

**Biological Prediction**

**Other**