PanGeA: Pan-Genome Annotation
Indexing Annotated Human Genome Collections

Andre Brehme, Sven Brümmer, Jonas Charfreitag, Jonas Ellert, Jannik Junghänel, Dominik Köppl, Christopher Osthues, Sven Rahmann, Dennis Rohde, Julian Sauer, Jonas Schmidt, Lars Schäpers, Uriel Elias Wiebelitz, Jens Zentgraf

Abstract
Motivation
- Novel high-throughput sequencing methods
- Huge sets of genomes with annotations
- Opportunities for personalized genome-based medicine
- Genomes of two individuals are up to 99% identical [FCS06]
- Genomes datasets too huge to maintain in commodity computers efficiently
- Opportunities for compression

Future Work
- Implement other approaches like grammar-based self-indexes [CN12]
- Use annotations for annotation based queries or more efficient query answering

Goal
Build an application, that
- Maintains a pangenome
- Answers queries like (approximate) pattern matching efficiently on the maintained pangenome

Implemented Data Structures
(a) Colored De Bruijn Graph [BBB+17] as a prefilter
(b) Journaled String Tree [RWR14]
(c) CHICO [Val16]
(d) Relative Lempel-Ziv [KPZ15]

Future Work
- Implement other approaches like grammar-based self-indexes [CN12]
- Use annotations for annotation based queries or more efficient query answering

Annotations
Genome 1
- BRCA2
- BKAP2
- MBL2

Annotated area (for specific gene) is very similar among all genomes
- Highly compressible areas can be found with help of annotations
- Useful to fix positions for sequence alignment

Index Features
Usage
1. Add/Delete genomes
2. Build implemented data structures (a) - (d)
3. Retrieve genomes or subsequences of genomes by ID and position or name of annotated area
4. Pattern Matching
5. Visualize the genome collection

(a) Colored De Bruijn Graph:
- Edges represent 4-k-1-mers
- AGG occurs in $s_2$ and $s_3$
- May return false positives due to circles
- Use DBG as a pre-filter
- Supports approximate pattern matching
- Succinct representation allows memory efficient implementation [BOSS12]

(b) Journaled String Trees
- Represent each genome as an order of copy operations that transform the reference genome into the other genome
- Save an insertion buffer for each genome and a tuple (v, p, l, r) for each copy operation
- Save tuples in balanced binary search tree

(c) CHICO (Compressed Hybrid Index for Repetitive Collections)
- Exact and approximate pattern matching
- Predefined maximum pattern length and maximum edit distance
- Utilizes RLZ-parsing to distinguish between two different possible match types
- Primary occurrence:
  - The pattern spans multiple RLZ-phrases
- Secondary occurrence:
  - The pattern is contained in a single RLZ-phrase

(d) Relative Lempel-Ziv
- User chooses reference genome $r$ (here $r = s_3$)
- Each genome is interpreted as a composition of reference genome substrings
- Each substring is stored as a (start position, length)-tuple
- Retrieval of each genome in linear time to the genome length

---

Position: 1 2 3 4 5 6 7 8 9 10
Genome $s_1$: TAGCOTTAGCA
Genome $s_2$: TAGGOTTAGCA
Genome $s_3$: TAGCOTTAGTA

PanGeA Index

---

Get (Sub-)sequence
- $s_1$: TAGCOTTAGCA
- $s_2$: TAGGOTTAGCA
- $s_3$: TAGCOTTAGTA

Search (AGC)
- Exact or Approximate Pattern Matching
- Pre-Filter (using (b))
- Apply Index Data Structures (using (b-d))
- Visualize