Hi-C interaction matrix correction using ICE in Rust

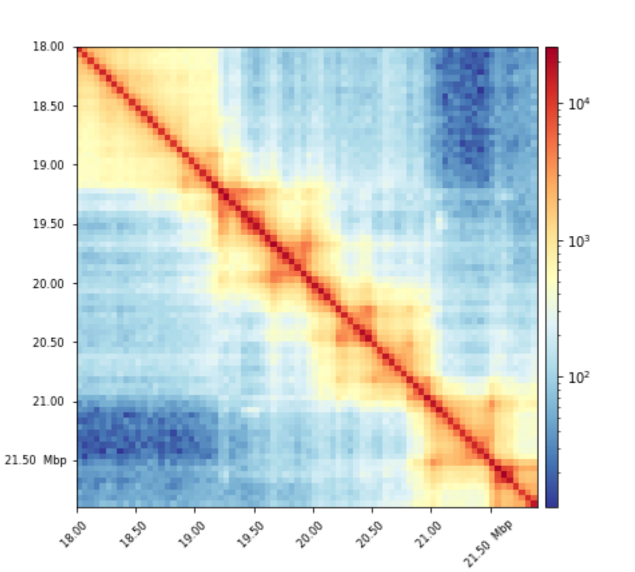
Bachelor thesis defense

Felix Karg

17. July 2019

University of Freiburg

Hi-C Contact Matrix



Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

High-Throughput 3C (Hi-C)

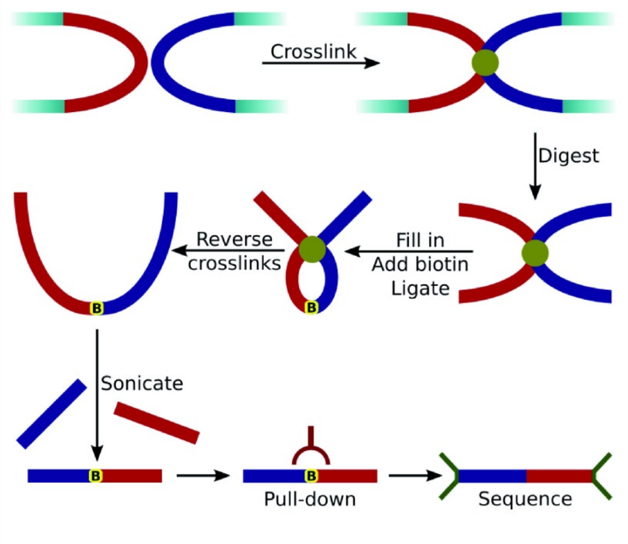


Image adapted from [1].

High-Throughput 3C (Hi-C)

Several biases:

High-Throughput 3C (Hi-C)

Several biases:

- Some regions are easier labeled with biotin than others

High-Throughput 3C (Hi-C)

Several biases:

- Some regions are easier labeled with biotin than others
- PCR artifacts [1]

High-Throughput 3C (Hi-C)

Several biases:

- Some regions are easier labeled with biotin than others
- PCR artifacts [1]
- Sequencing itself has multiple biases [2]

High-Throughput 3C (Hi-C)

Several biases:

- Some regions are easier labeled with biotin than others
- PCR artifacts [1]
- Sequencing itself has multiple biases [2]
- Where to map multiple/unclear reads

HiCExplorer

Helpful tools, especially for:

HiCExplorer

Helpful tools, especially for:

- Data correction

HiCExplorer

Helpful tools, especially for:

- Data correction
- Analysis

HiCExplorer

Helpful tools, especially for:

- Data correction
- Analysis
- Visualization

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

ICE as described in Imakaev et al. 2012 [3]

ICE as described in Imakaev et al. 2012 [3]

- O_{ij}: raw data

ICE as described in Imakaev et al. 2012 [3]

- O_{ij}: raw data
- T_{ij}: relative contact probabilities

ICE as described in Imakaev et al. 2012 [3]

- O_{ij}: raw data
- T_{ij}: relative contact probabilities
- B_i, B_j: cumulative biases

ICE as described in Imakaev et al. 2012 [3]

Goal: Obtain B and T_{ij}.

ICE as described in Imakaev et al. 2012 [3]

Goal: Obtain B and T_{ij}.

Achieve this by explicitly solving:

$$O_{ij} = B_i B_j T_{ij} \tag{1}$$

$$\sum_{i=1,|i-j|>1}^{N} T_{ij} = 1 \tag{2}$$

ICE as described in Imakaev et al. 2012 [3]

$$T_{ij} = \begin{bmatrix} d & d_{+1} & t_{1,3} & \dots & \dots & t_{1,n} \\ d_{-1} & d & d_{+1} & \dots & \dots & t_{2,n} \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ t_{n-1,1} & \dots & \dots & d_{-1} & d & d_{+1} \\ t_{n,1} & \dots & \dots & t_{n,n-2} & d_{-1} & d \end{bmatrix}$$

$$\sum_{i=1,|i-j|>1}^{N} T_{ij} = 1$$

ICE as described in Imakaev et al. 2012 [3]

$$T_{ij} = \begin{bmatrix} d & d_{+1} & t_{1,3} & \dots & \dots & t_{1,n} \\ d_{-1} & d & d_{+1} & \dots & \dots & t_{2,n} \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ t_{n-1,1} & \dots & \dots & d_{-1} & d & d_{+1} \\ t_{n,1} & \dots & \dots & t_{n,n-2} & d_{-1} & d \end{bmatrix}$$

$$\sum_{i=1,|i-j|>1}^{N} T_{ij} = 1$$

ICE as described in Imakaev et al. 2012 [3]

$$T_{ij} = \begin{bmatrix} d & d_{+1} & t_{1,3} & \dots & \dots & t_{1,n} \\ d_{-1} & d & d_{+1} & \dots & \dots & t_{2,n} \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ t_{n-1,1} & \dots & \dots & d_{-1} & d & d_{+1} \\ t_{n,1} & \dots & \dots & t_{n,n-2} & d_{-1} & d \end{bmatrix}$$

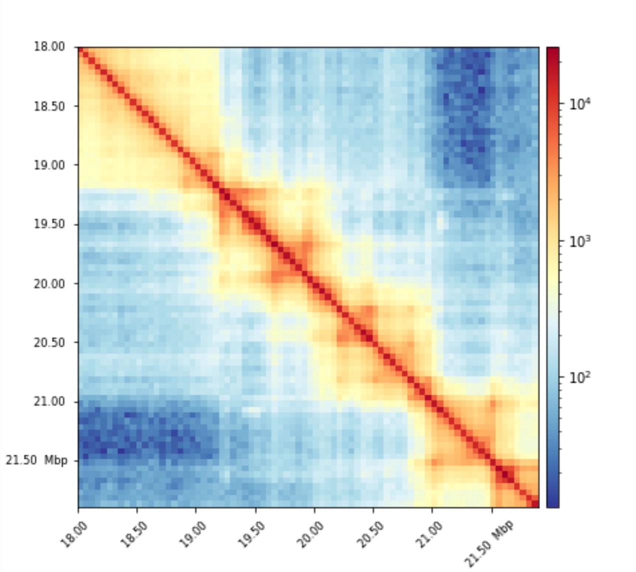
$$\sum_{i=1,|i-j|>1}^{N} T_{ij} = 1$$

ICE as described in Imakaev et al. 2012 [3]

$$T_{ij} = \begin{bmatrix} d & d_{+1} & t_{1,3} & \dots & \dots & t_{1,n} \\ d_{-1} & d & d_{+1} & \dots & \dots & t_{2,n} \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ t_{n-1,1} & \dots & \dots & d_{-1} & d & d_{+1} \\ t_{n,1} & \dots & \dots & t_{n,n-2} & d_{-1} & d \end{bmatrix}$$

$$\sum_{i=1,|i-j|>1}^{N} T_{ij} = 1$$

ICE as described in Imakaev et al. 2012 [3]



Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

Rust: Language Features

Rust: Language Features

- Syntax seems similar to C/C++

Rust: Language Features

- Syntax seems similar to C/C++
- Semantically closer to ML/Haskell

Rust: Language Features

- Syntax seems similar to C/C++
- Semantically closer to ML/Haskell
- Memory safety (no NULL)

Rust: Language Features

- Syntax seems similar to C/C++
- Semantically closer to ML/Haskell
- Memory safety (no NULL)
- Ownership

Rust: Language Features

- Syntax seems similar to C/C++
- Semantically closer to ML/Haskell
- Memory safety (no NULL)
- Ownership
- Borrowing

Rust: Language Features

- Syntax seems similar to C/C++
- Semantically closer to ML/Haskell
- Memory safety (no NULL)
- Ownership
- Borrowing
- Lifetimes

Rust: Comparison

Speed comparison	C	Rust	C++
n-body	7.49	**5.72**	8.18
binary-trees	3.48	**3.15**	3.79
pidigits	**1.75**	**1.75**	1.89
reverse-complement	1.78	1.61	**1.55**
spectral-norm	1.98	**1.97**	1.98
fannkuch-redux	**8.61**	10.23	10.08
k-nucleotide	5.01	5.25	**3.76**
fasta	1.36	1.47	**1.33**
mandelbrot	1.65	1.96	**1.50**
regex-redux	**1.46**	2.43	1.82
Fastest in:	3/10	4/10	4/10

Runtime measured in **seconds**. Numbers for C from [4] and for C++ from [5]. Both show the same numbers for Rust.

Rust: Advantages and Disadvantages

General advantages:

Rust: Advantages and Disadvantages

General advantages:

- High-Level features

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling
- Strong type system

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling
- Strong type system

General disadvantages:

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling
- Strong type system

General disadvantages:

- Young ecosystem

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling
- Strong type system

General disadvantages:

- Young ecosystem
- Steep learning curve

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling
- Strong type system

General disadvantages:

- Young ecosystem
- Steep learning curve
- Higher initial compile times

Rust: Advantages and Disadvantages

General advantages:

- High-Level features
- Fast language
- Safe memory handling
- Strong type system

General disadvantages:

- Young ecosystem
- Steep learning curve
- Higher initial compile times
- Language features not yet available

Rust: Advantages and Disadvantages

Advantages for this project:

Rust: Advantages and Disadvantages

Advantages for this project:

- Own CSRMatrix implementation

Rust: Advantages and Disadvantages

Advantages for this project:

- Own CSRMatrix implementation
- Not many dependencies

Rust: Advantages and Disadvantages

Advantages for this project:

- Own CSRMatrix implementation
- Not many dependencies
- Faster and smaller, very specific

Rust: Advantages and Disadvantages

Advantages for this project:

- Own CSRMatrix implementation
- Not many dependencies
- Faster and smaller, very specific

Disadvantages for this project:

Rust: Advantages and Disadvantages

Advantages for this project:

- Own CSRMatrix implementation
- Not many dependencies
- Faster and smaller, very specific

Disadvantages for this project:

- No general implementation of CSRMatrix

Rust: Advantages and Disadvantages

Advantages for this project:

- Own CSRMatrix implementation
- Not many dependencies
- Faster and smaller, very specific

Disadvantages for this project:

- No general implementation of CSRMatrix
- Only subset of features when compared to SciPy implementation

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

API: Rust to Python

There are three main ways to use Rust from Python:

API: Rust to Python

There are three main ways to use Rust from Python:

- rust-cpython

API: Rust to Python

There are three main ways to use Rust from Python:

- rust-cpython
- pyO3

API: Rust to Python

There are three main ways to use Rust from Python:

- rust-cpython
- pyO3
- dylib

rust-cpython

rust-cpython

- Similar to C-Python headers

rust-cpython

- Similar to C-Python headers
- Grants access to the Python GIL

rust-cpython

- Similar to C-Python headers
- Grants access to the Python GIL
- Renaming needed

rust-cpython

- Similar to C-Python headers
- Grants access to the Python GIL
- Renaming needed
- Stable Rust

rust-cpython

- Similar to C-Python headers
- Grants access to the Python GIL
- Renaming needed
- Stable Rust
- Easy package creation

pyO3

pyO3

- Fork from rust-cpython

pyO3

- Fork from rust-cpython
- Grants access to the Python GIL

pyO3

- Fork from rust-cpython
- Grants access to the Python GIL
- Renaming needed

pyO3

- Fork from rust-cpython
- Grants access to the Python GIL
- Renaming needed
- Nightly Rust

pyO3

- Fork from rust-cpython
- Grants access to the Python GIL
- Renaming needed
- Nightly Rust
- Easy package creation

dylib

dylib

- Recommended in the Rust book

dylib

- Recommended in the Rust book
- No renaming needed

dylib

- Recommended in the Rust book
- No renaming needed
- Stable Rust

dylib

- Recommended in the Rust book
- No renaming needed
- Stable Rust
- No access to the Python GIL

dylib

- Recommended in the Rust book
- No renaming needed
- Stable Rust
- No access to the Python GIL
- Package creation possible

API Comparison for Rust in Python

API Comparison	rust-cpython	pyO3	dylib
Memory from Python	Yes	Yes	**Optional**
Renaming needed	Yes	Yes	**No**
Stable Rust	**Yes**	No	**Yes**
Platform-specific	Yes	Yes	**No**
Implementation effort	Medium	Medium	**Low**
Creating python packages	**Easy**	**Easy**	Normal
Good in:	2/6	1/6	**5/6**

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

Compared implementations

Compared implementations

- 'ICE' - Python implementation

Compared implementations

- 'ICE' - Python implementation
- 'KR' - C++ implementation

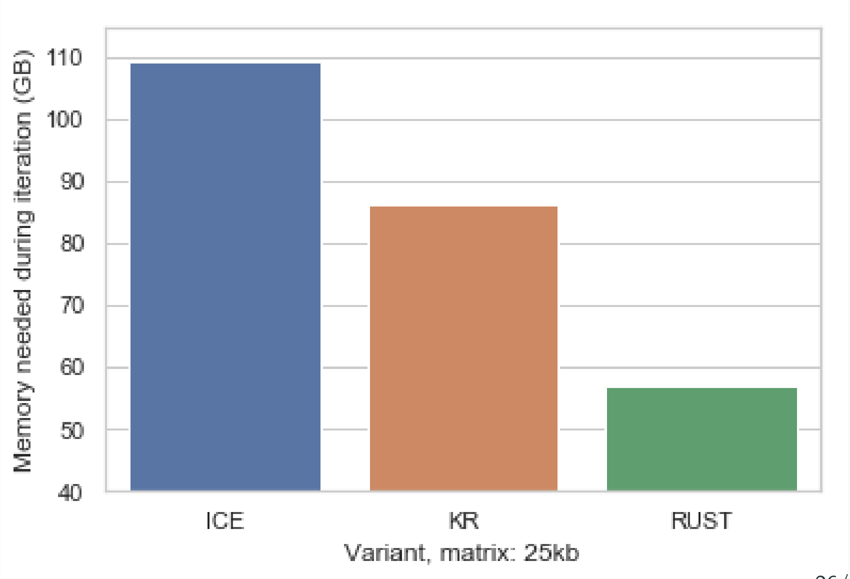
Compared implementations

- 'ICE' - Python implementation
- 'KR' - C++ implementation
- 'RUST' - this implementation

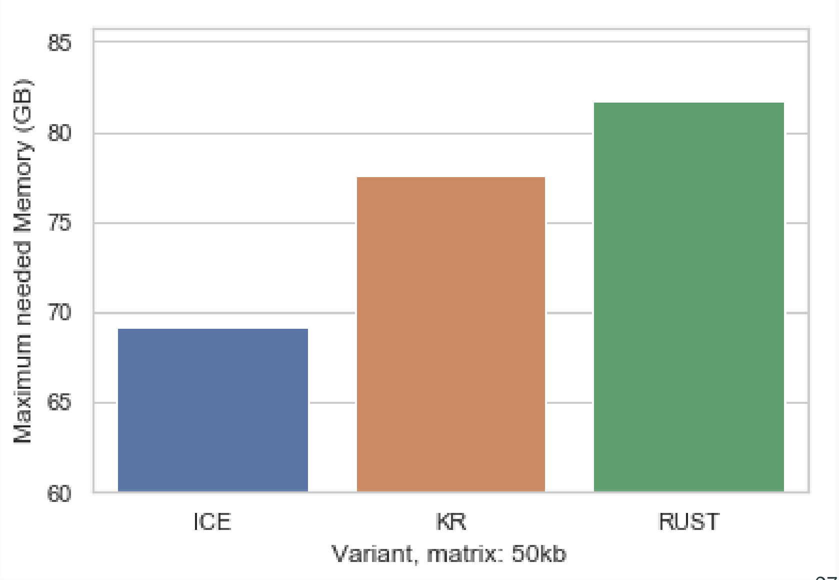
Data for Testing

Name	25kb	50kb
Filesize	1.1 GByte	0.73 GByte
Size	123,841	61,928
Bin length	25,000	50,000
Non-zero elements	1,530,533,003	1,053,216,825

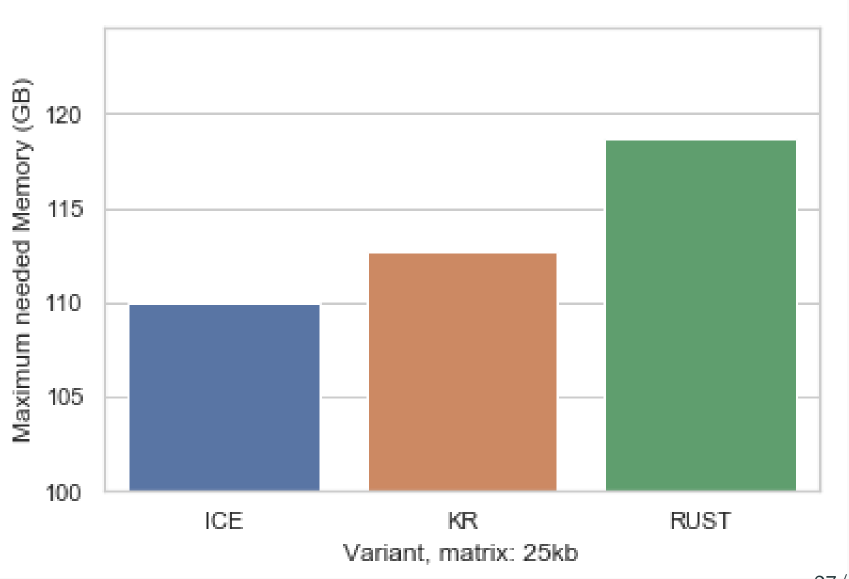
Memory Requirements during correction



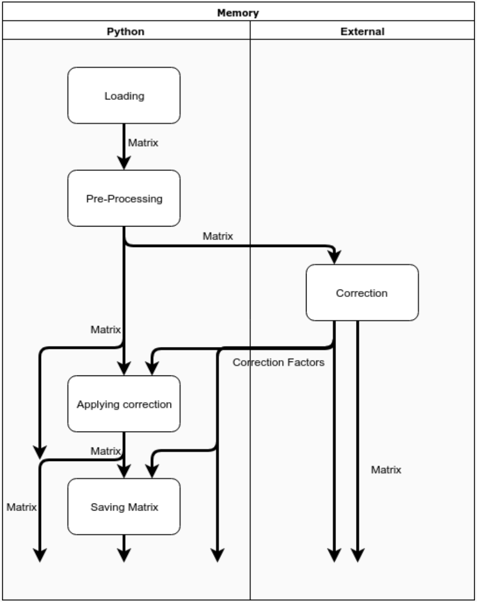
Peak Memory Usage



Peak Memory Usage



Peak Memory Usage

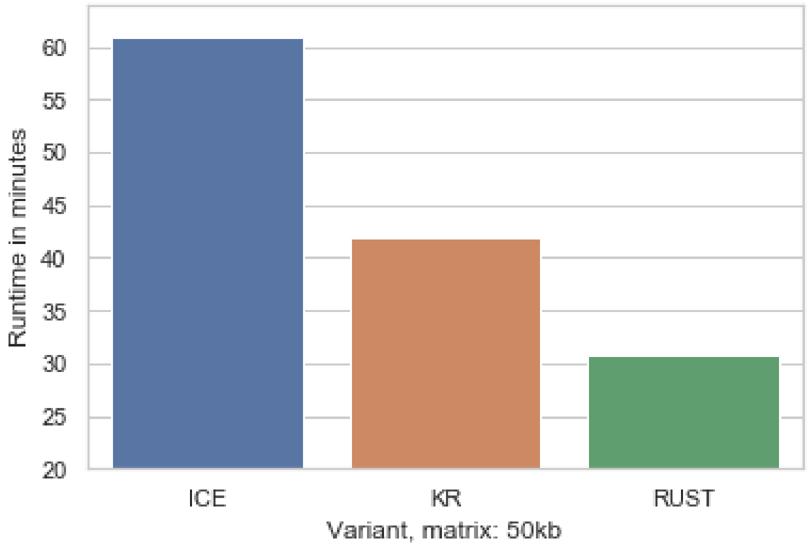


Comparison of Memory needs

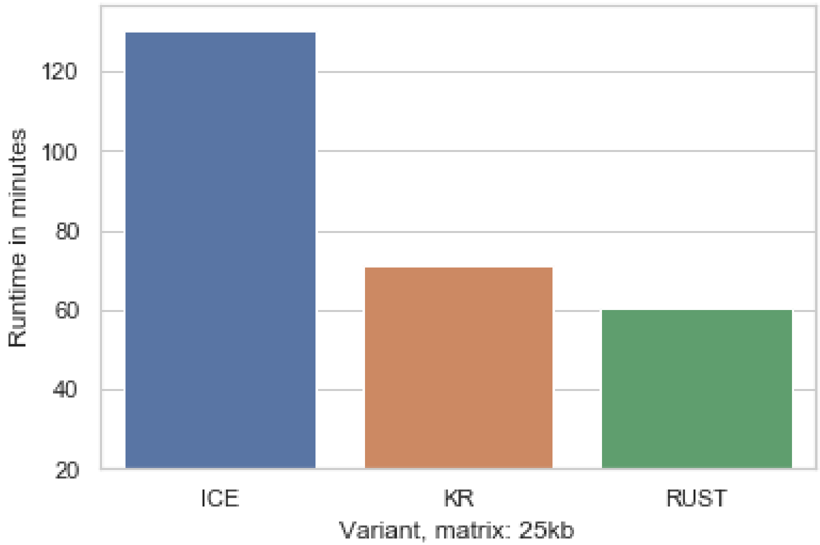
Memory needs Comparison	ICE	KR	RUST
During correction (50kb)	54.6	43.1	**39.0**
Maximum (50kb)	**69.2**	77.6	81.7
During correction (25kb)	110.0	86.0	**57.0**
Maximum (25kb)	**110.0**	112.7	118.6

Runtime Length

Runtime Length

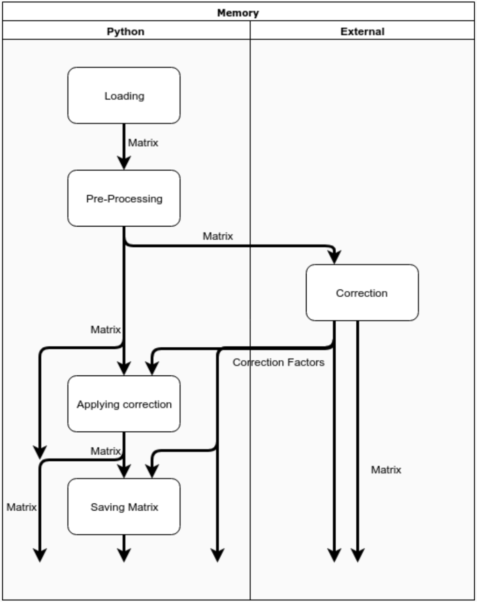


Runtime Length



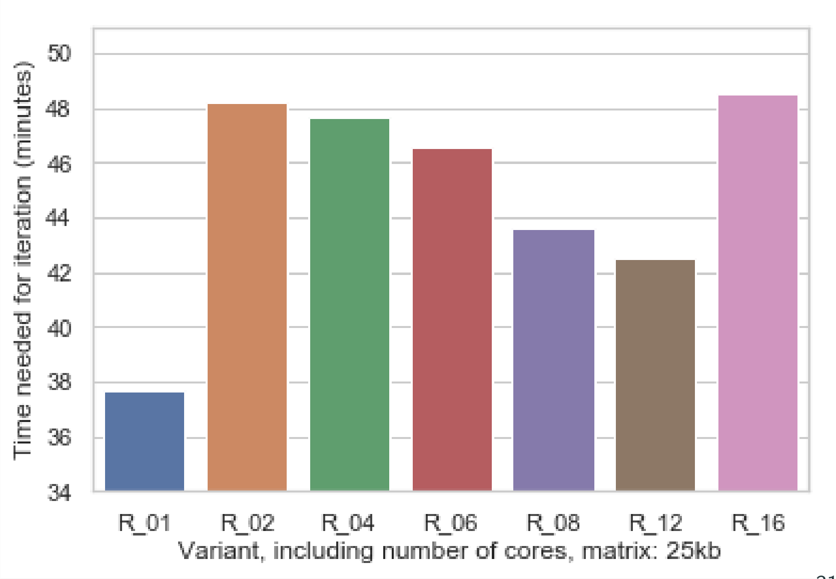
Control Flow Diagram

Control Flow Diagram

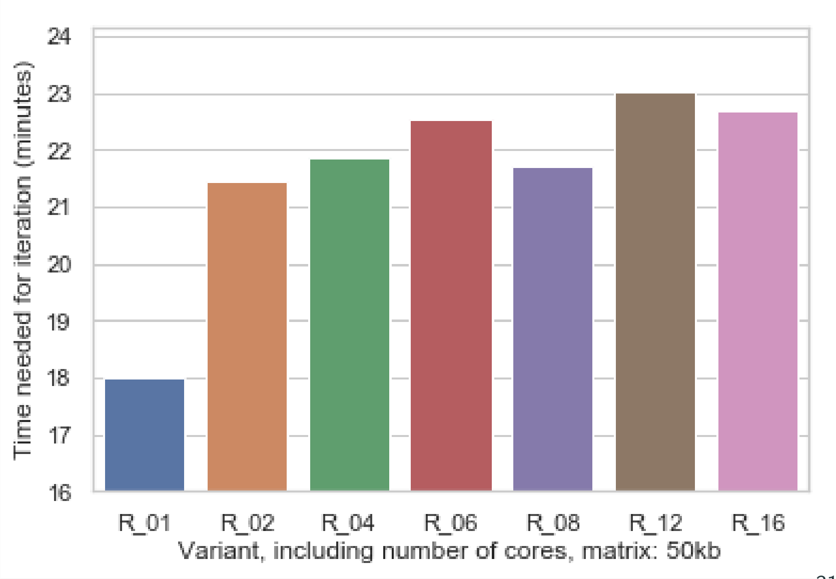


Multicore Runtime Length Comparison

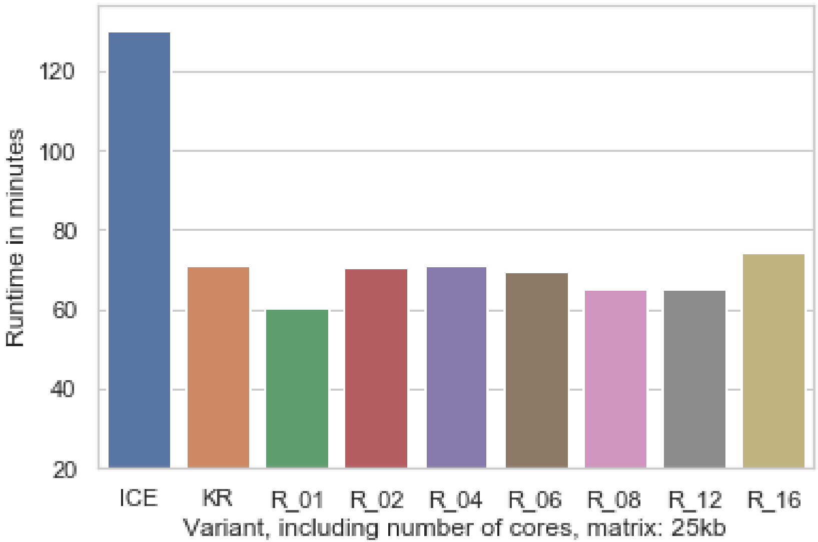
Multicore Runtime Length Comparison



Multicore Runtime Length Comparison

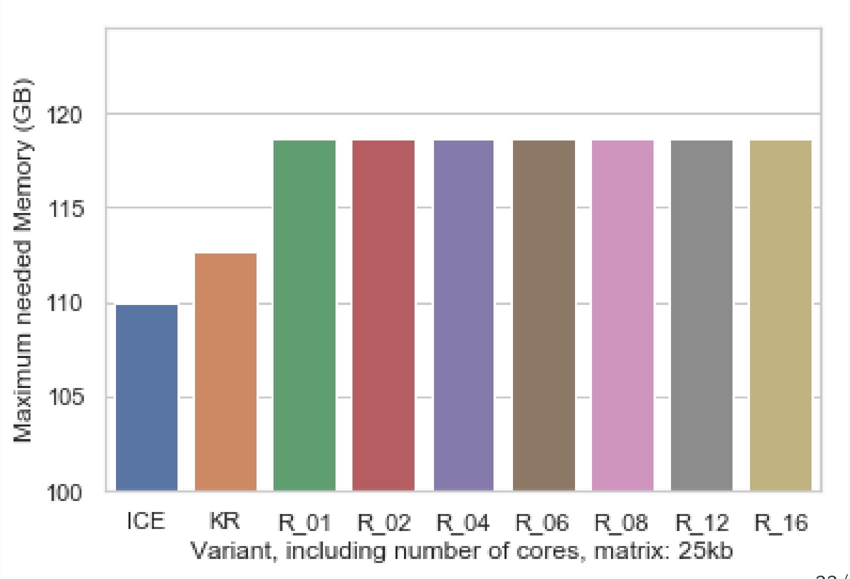


Overall Runtime Length Comparison

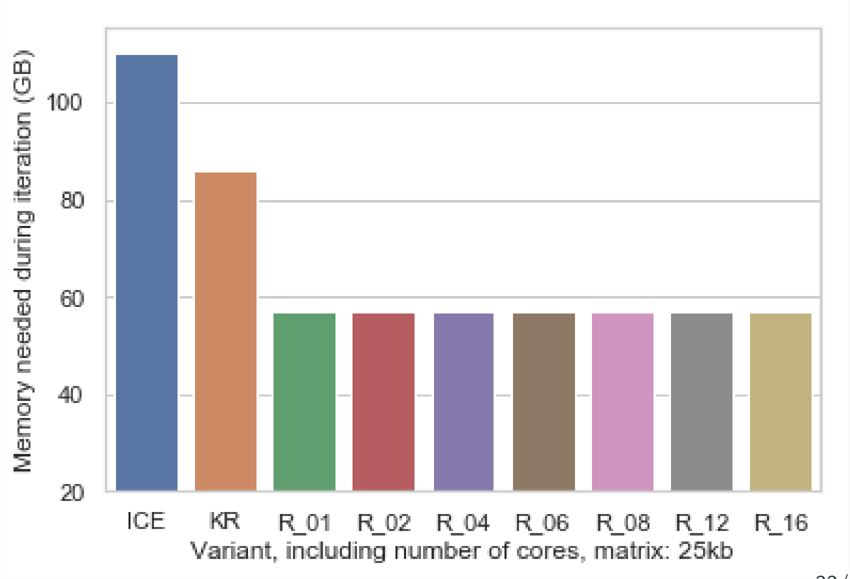


Multicore Memory Comparison

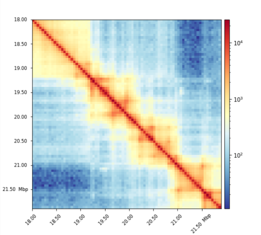
Multicore Memory Comparison



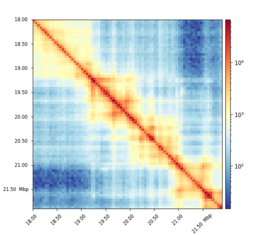
Multicore Memory Comparison



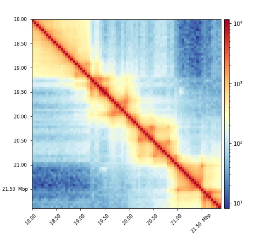
Comparison of Results



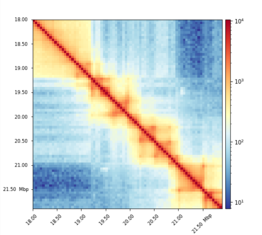
(a) Uncorrected



(b) RUST



(c) KR



(d) ICE

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

Conclusion Regarding the Integration of RUST

Conclusion Regarding the Integration of RUST

- Test the alternatives more

Conclusion Regarding the Integration of RUST

- Test the alternatives more
- Better integration should be possible

Conclusion Regarding the Integration of RUST

- Test the alternatives more
- Better integration should be possible
- All in all: Went better than expected

Conclusion Regarding Computation Comparisons

Conclusion Regarding Computation Comparisons

- Reduction of memory usage during correction achieved

Conclusion Regarding Computation Comparisons

- Reduction of memory usage during correction achieved
- Reduction of runtime achieved

Conclusion Regarding Computation Comparisons

- Reduction of memory usage during correction achieved
- Reduction of runtime achieved
- Parallelism does not offer significant benefits yet

Remaining Questions

Remaining Questions

- Writing code for faster parallelism

Remaining Questions

- Writing code for faster parallelism
- Speedup when parallelizing more

Remaining Questions

- Writing code for faster parallelism
- Speedup when parallelizing more
- Pure implementations needing even less memory?

Remaining Questions

- Writing code for faster parallelism
- Speedup when parallelizing more
- Pure implementations needing even less memory?
- Is KR faster for bigger matrices?

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Results

Conclusion

Sources

Sources i

- S. Wingett, P. Ewels, M. Furlan-Magaril, T. Nagano, S. Schoenfelder, P. Fraser, and S. Andrews, "Hicup: pipeline for mapping and processing hi-c data," *F1000Research*, vol. 4, 2015.
- D. Aird, M. G. Ross, W.-S. Chen, M. Danielsson, T. Fennell, C. Russ, D. B. Jaffe, C. Nusbaum, and A. Gnirke, "Analyzing and minimizing pcr amplification bias in illumina sequencing libraries," *Genome biology*, vol. 12, no. 2, p. R18, 2011.

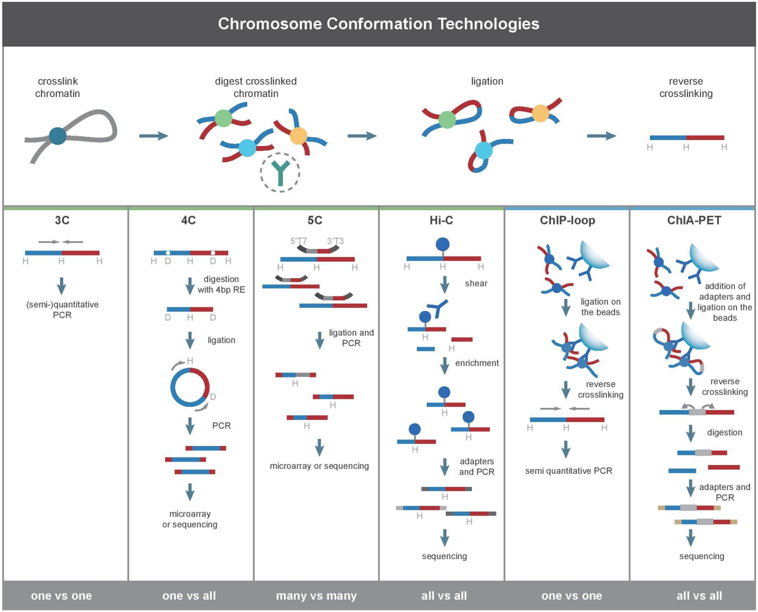
Sources ii

- M. Imakaev, G. Fudenberg, R. P. McCord, N. Naumova, A. Goloborodko, B. R. Lajoie, J. Dekker, and L. A. Mirny, "Iterative correction of hi-c data reveals hallmarks of chromosome organization," *Nature methods*, vol. 9, no. 10, p. 999, 2012.
- "Rust comparison with c." `https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/rust.html`, 2019. **accessed 2019-06-26.**

Sources iii

- "Rust comparison with c++." `https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/rust-gpp.html`, 2019. **accessed 2019-06-26.**
- G. Li, L. Cai, H. Chang, P. Hong, Q. Zhou, E. V. Kulakova, N. A. Kolchanov, and Y. Ruan, "Chromatin interaction analysis with paired-end tag (chia-pet) sequencing technology and application," *BMC Genomics*, vol. 15, p. S11, Dec 2014.

End



Chromosome Conformation Technologies
crosslink chromatin
digest crosslinked chromatin
ligation
reverse crosslinking
H H H
3C
4C
5C
Hi-C
ChIP-loop
ChIA-PET
(semi-)quantitative PCR
digestion with 4bp RE
ligation
PCR
microarray or sequencing
5'T7
3'T3
ligation and PCR
microarray or sequencing
shear
enrichment
adapters and PCR
sequencing
ligation on the beads
reverse crosslinking
semi quantitative PCR
addition of adapters and ligation on the beads
reverse crosslinking
digestion
adapters and PCR
sequencing
one vs one
one vs all
many vs many
all vs all
one vs one
all vs all

Image from [6].

ICE as described in Imakaev et al. 2012 [3]

Each iteration, compute:

ICE as described in Imakaev et al. 2012 [3]

Each iteration, compute:

$$S_i = \sum_j W_{ij} \tag{3}$$

$$\Delta B_i = S_i / mean(S) \tag{4}$$

ICE as described in Imakaev et al. 2012 [3]

Each iteration, compute:

$$S_i = \sum_j W_{ij} \tag{3}$$

$$\Delta B_i = S_i / mean(S) \tag{4}$$

$$W_{ij} = W_{ij} / \Delta B_i \Delta B_j \tag{5}$$

$$B_i = B_i \cdot \Delta B_i \tag{6}$$

Rust: Code Example

Code Example 1

```rust
fn main() {
    let mut v = vec![];     // ---|
    v.push("Hello");        // <--|
                            //    |
    let x = &v[0];          // -| |
                            //  | |
                            //  | |
    v.push("world");        // <X-|
    println!("{}", x);      // -| |
}                           // ---|
```

Rust: Code Example

Output Nr. 1

```
error[E0502]: cannot borrow `v` as mutable because it is
also borrowed as immutable
 --> src/main.rs:5:5
  |
5 |     let x = &v[0];
  |             - immutable borrow occurs here
8 |     v.push("world");
  |     ^^^^^^^^^^^^^^^ mutable borrow occurs here
9 |     println!("{}", x);
  |                    - immutable borrow later used here
```

Rust: Code Example

Code Example 2

```rust
fn main() {
    let mut v = vec![];      // ---|
    v.push("Hello");         // <--|
                             //    |
    let x = &v[0];           // -| |
    println!("{}", x);       // -| |
                             //    |
    v.push("world");         // <--|
    println!("{}", v[1]);    // <--|
}                            // ---|
```

Rust: Code Example

Output Nr. 2

```
Hello
world
```

Test Server Specification

Virtual Server Specification	
Available Cores / Threads	16 / 32
Working Memory (RAM)	120 GByte

Processor Specification	
Processor	Intel® Xeon® E5-2630V4
Number of Cores/Threads	10 / 20
Base/Turbo frequency	2.2 GHz / 3.1 GHz