Hi-C interaction matrix correction using ICE in Rust

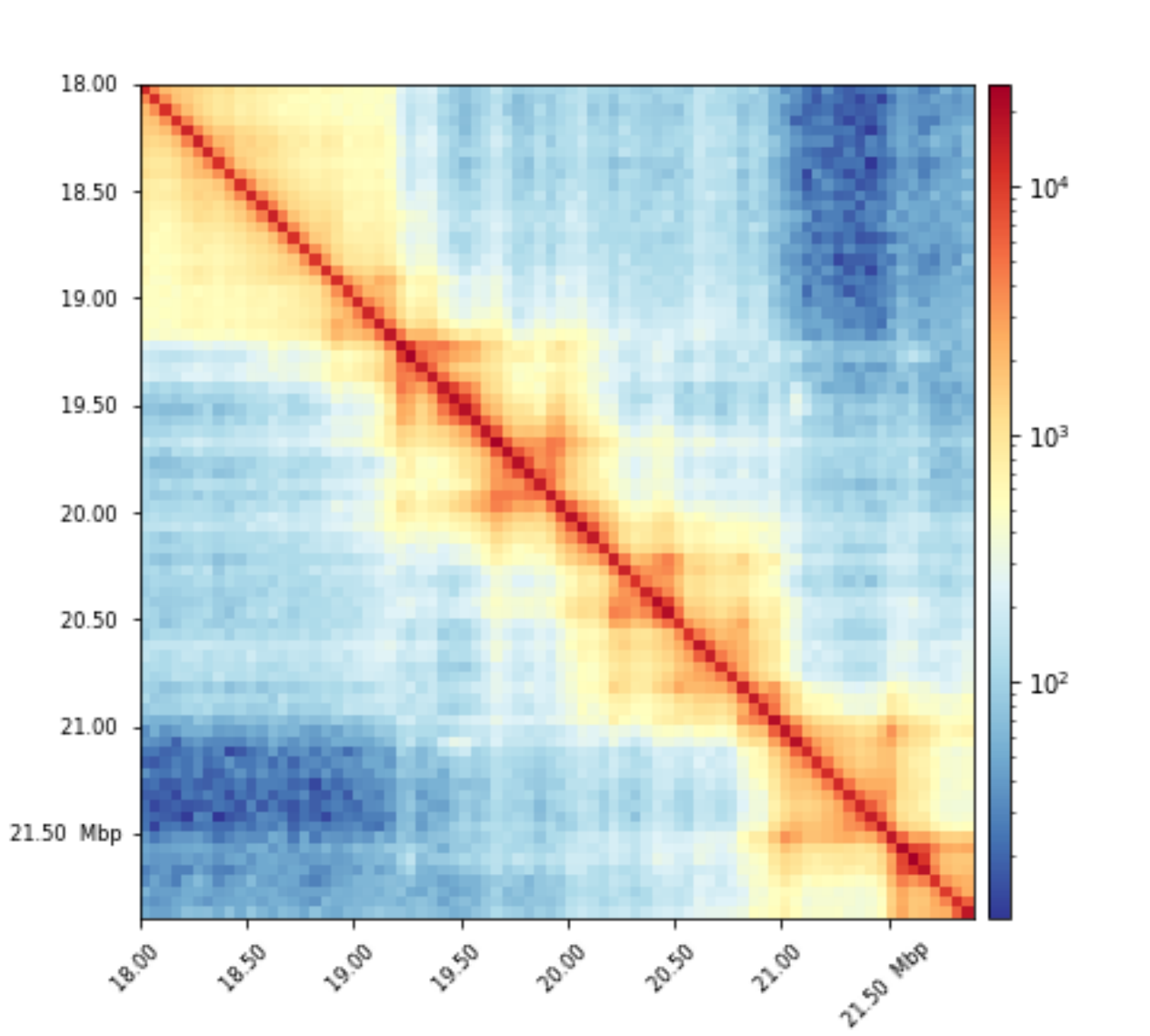
Bachelor thesis defense

Felix Karg

17. Juli 2019

University of Freiburg

Hi-C Contact Matrix



Content

Hi-C

ICE

Rust

Integration of Rust in Python

Comparison

Conclusion

Sources

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Comparison

Conclusion

Sources

High-Throughput 3C (Hi-C)

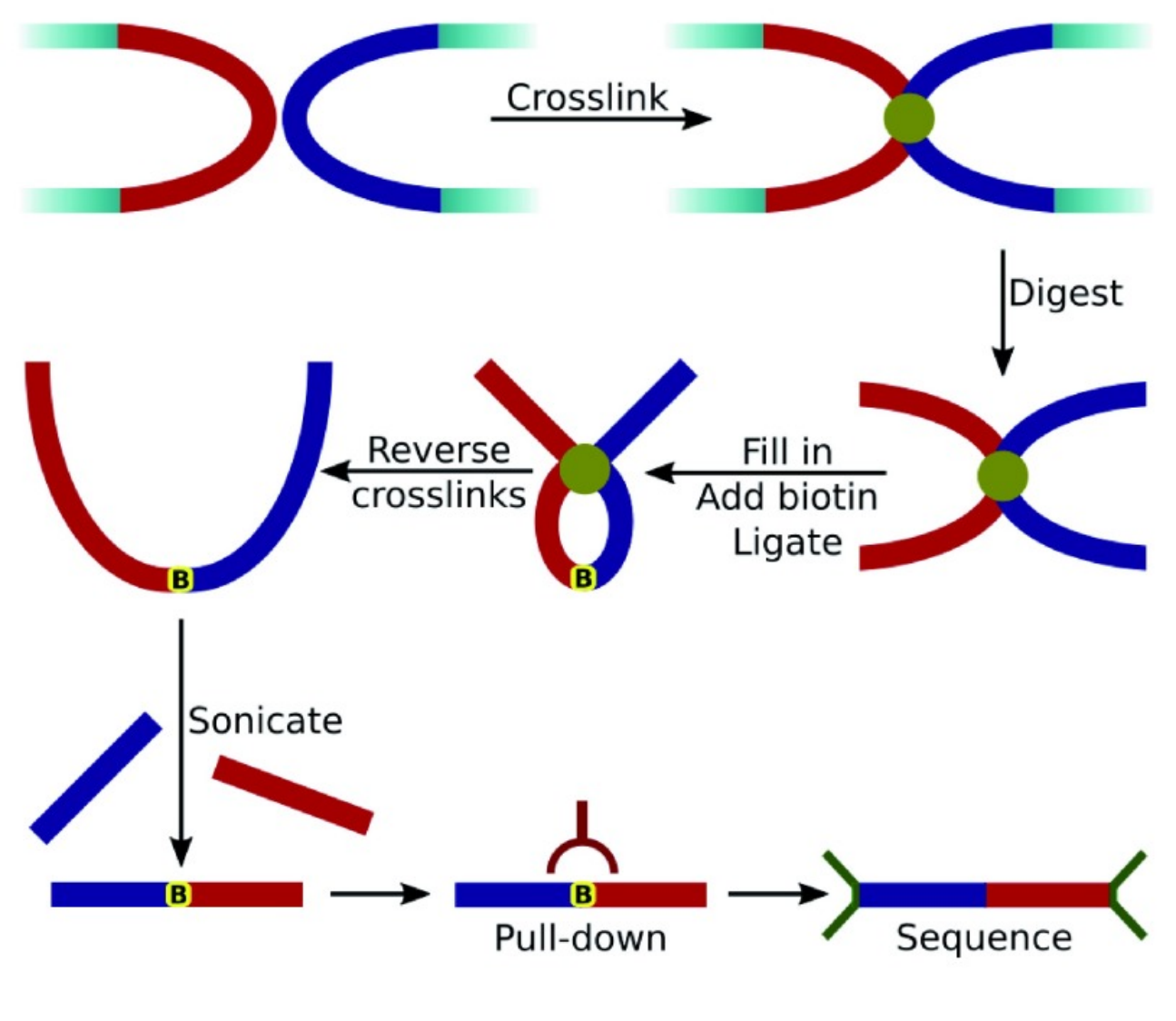


Image adapted from [1]

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

Comparison

Conclusion

Sources

ICE as described in Imakaev et al. 2012 [2]

ICE as described in Imakaev et al. 2012 [2]

- O_{ij}: raw data

ICE as described in Imakaev et al. 2012 [2]

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

ICE as described in Imakaev et al. 2012 [2]

- O_{ij}: raw data
- T_{ij}: relative contact probabilities
- W_{ij}: working copy of O_{ij}, becoming T_{ij}

ICE as described in Imakaev et al. 2012 [2]

- O_{ij}: raw data
- T_{ij}: relative contact probabilities
- W_{ij}: working copy of O_{ij}, becoming T_{ij}
- S_i: sum of row i of W_{ij}

ICE as described in Imakaev et al. 2012 [2]

- O_{ij}: raw data
- T_{ij}: relative contact probabilities
- W_{ij}: working copy of O_{ij}, becoming T_{ij}
- S_i: sum of row i of W_{ij}
- B_i, B_j: cumulative biases

ICE as described in Imakaev et al. 2012 [2]

Goal: Obtain B and T_{ij}.

ICE as described in Imakaev et al. 2012 [2]

Goal: Obtain B and T_{ij}.

Do 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 [2]

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

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

ICE as described in Imakaev et al. 2012 [2]

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

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

ICE as described in Imakaev et al. 2012 [2]

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

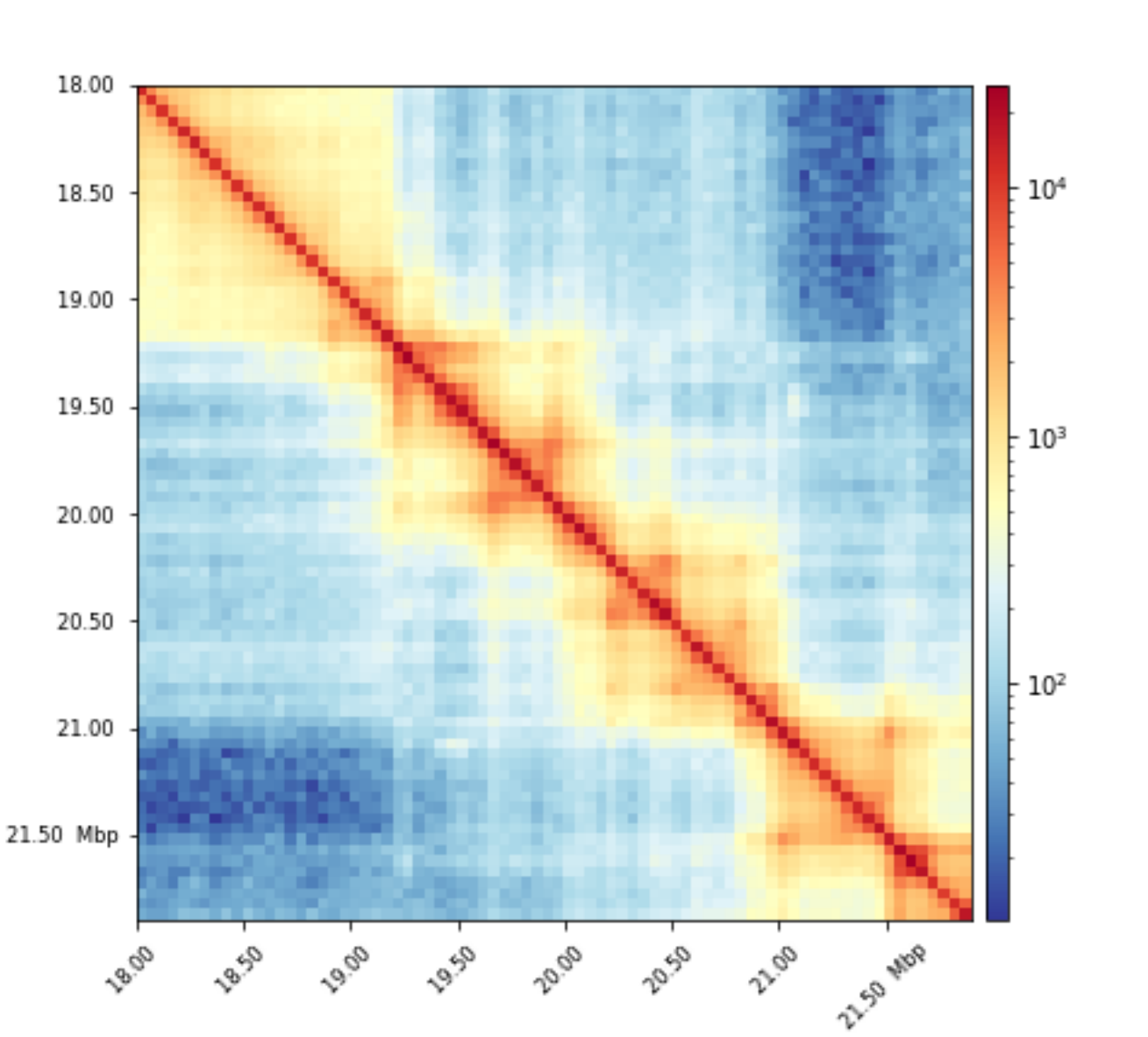
$$\sum_{i=1,|i-j|>1}^{N} T_{ij} = 1$$

ICE as described in Imakaev et al. 2012 [2]

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

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

ICE as described in Imakaev et al. 2012 [2]



ICE as described in Imakaev et al. 2012 [2]

Each iteration, compute:

ICE as described in Imakaev et al. 2012 [2]

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 [2]

Each iteration, compute:

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

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

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

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

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Comparison

Conclusion

Sources

Rust: Short History

Rust: Short History

- Started out in 2006

Rust: Short History

- Started out in 2006
- Mozilla started sponsoring in 2009

Rust: Short History

- Started out in 2006
- Mozilla started sponsoring in 2009
- Self-compiled in 2011

Rust: Short History

- Started out in 2006
- Mozilla started sponsoring in 2009
- Self-compiled in 2011
- 1.0 released in 2015

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)
- Memory Management (RAII)

Rust: Language Features

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

Rust: Language Features

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

Rust: Language Features

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

Rust: Comparison

Speed coparison	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	**1.5**
regex-redux	**1.46**	2.43	1.82
Fastest in:	3/10	4/10	4/10

Runtime measured in **seconds**. Numbers for C from [3] and for C++ from [4]. Both show the same numbers for Rust.

Rust: Advantages

In General:

Rust: Advantages

In General:

- High-Level Features

Rust: Advantages

In General:

- High-Level Features
- Fast Language

Rust: Advantages

In General:

- High-Level Features
- Fast Language
- Safe Memory handling

Rust: Advantages

In General:

- High-Level Features
- Fast Language
- Safe Memory handling
- Strong Type system

Rust: Advantages

For this Project:

Rust: Advantages

For this Project:

- Own CSRMatrix implementation

Rust: Advantages

For this Project:

- Own CSRMatrix implementation
- Faster and smaller, very specific

Rust: Disadvantages

In General:

Rust: Disadvantages

In General:

- Young ecosystem

Rust: Disadvantages

In General:

- Young ecosystem
- Steep learning curve

Rust: Disadvantages

In General:

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

Rust: Disadvantages

In General:

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

Rust: Disadvantages

For this Project:

Rust: Disadvantages

For this Project:

- No general implementation of CSRMatrix

Rust: 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

Comparison

Conclusion

Sources

Integration of Rust in Python

Konkrete Fragen beantworten: Integration Rust ¡-¿ Python

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Comparison

Conclusion

Sources

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Comparison

Conclusion

Sources

Content

Hi-C

ICE

Rust

Integration of Rust in Python

Comparison

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.
- 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.

Sources ii

- "Rust comparison with c." `https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/rust.html`, 2019. **accessed 2019-06-26.**
- "Rust comparison with c++." `https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/rust-gpp.html`, 2019. **accessed 2019-06-26.**

Rust: Code Example

Code Example 1

```
1  fn main() {
2      let mut v = vec![];        // ---|
3      v.push("Hello");           // <--|
4                                 //    |
5      let x = &v[0];             // -| |
6                                 //  | |
7                                 //  | |
8      v.push("world");           // <X-|
9      println!("{}", x);         // -| |
10 }                              // ---|
```

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

```
1  fn main() {
2      let mut v = vec![];        // ---|
3      v.push("Hello");           // <--|
4                                 //    |
5      let x = &v[0];             // -| |
6      println!("{}", x);         // -| |
7                                 //    |
8      v.push("world");           // <--|
9      println!("{}", v[1]);      // <--|
10 }                              // ---|
```

Rust: Code Example

Output Nr. 2

```
Hello
world
```