

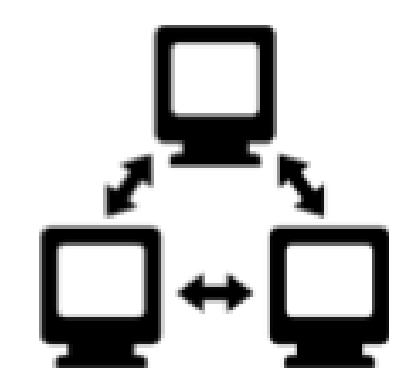
Introduction

- It has been twelve years since the publication of the ABySS short-read *de novo* genome assembler, and four since its successor, ABySS 2.



2009

The first ABySS release utilized a de Bruijn Graph distributed across multiple machines. This allowed for memory aggregation required to assemble large genomes such as *H. sapiens* at the time.

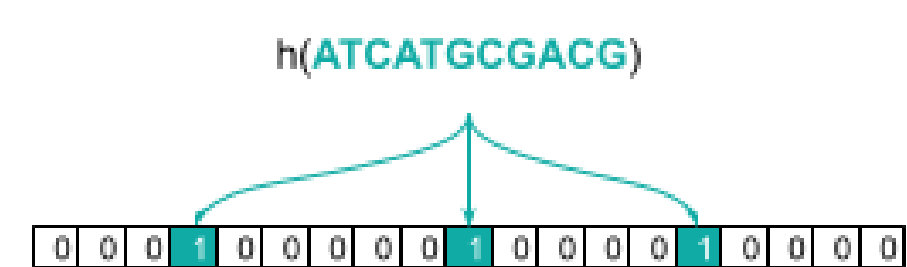


Intermachine communication has extra overhead and requires infrastructure. This motivated the design of ABySS 2.

2017

ABYSS v2

ABySS 2 employed a Bloom filter in order to decrease memory usage up to 10-fold

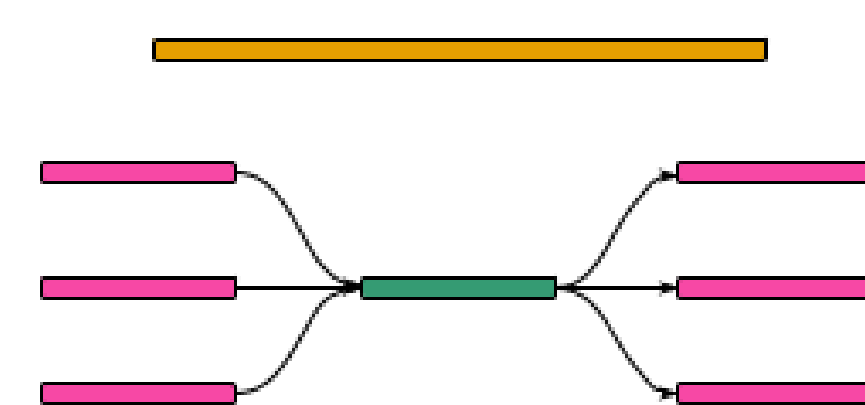


Low memory footprint enabled assembly of large genomes even on commodity hardware. Here, a multithreaded model on a single machine is used over distributed computing.

ABYSS 2.5

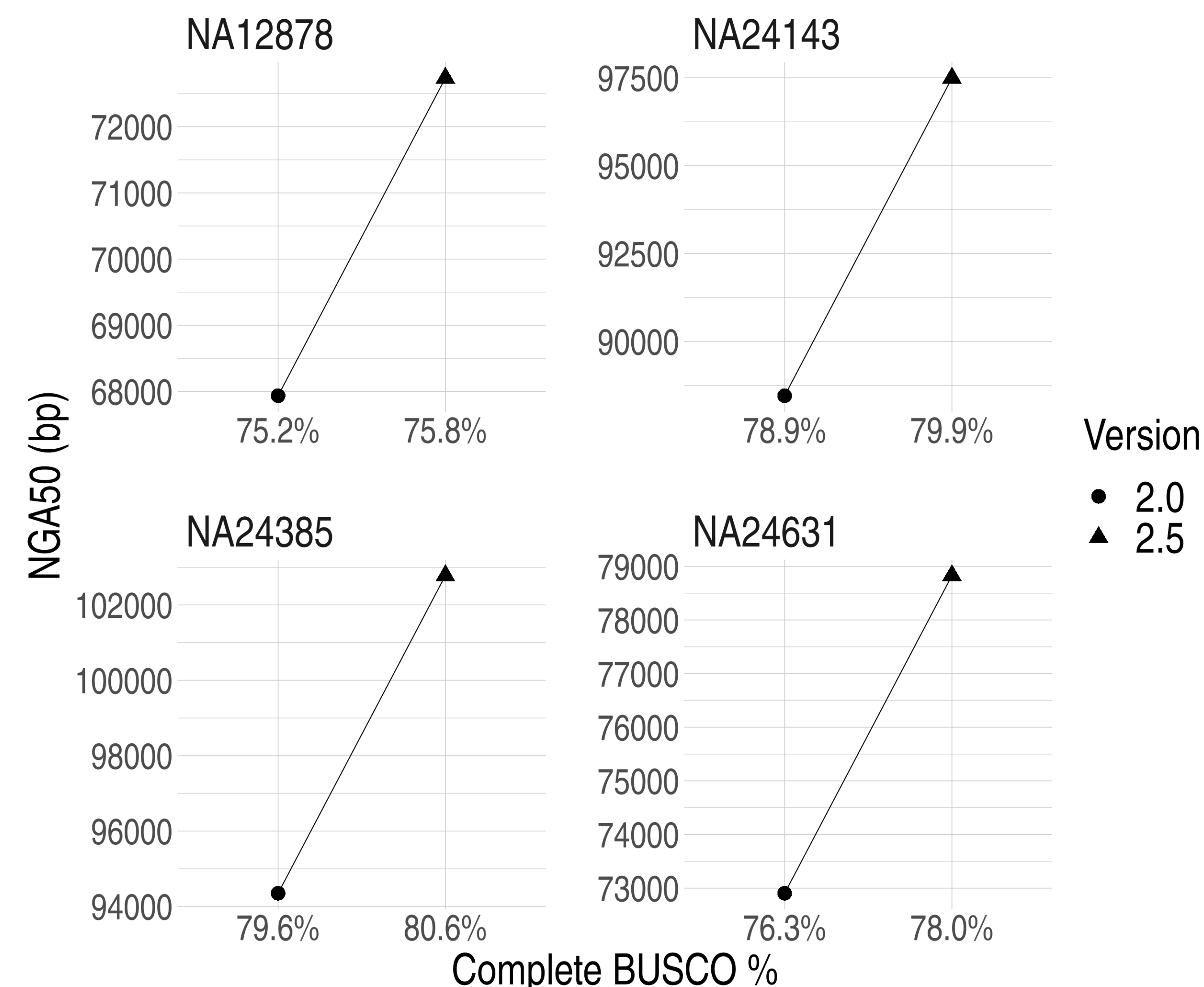
2021

ABySS 2.5 further improves the assembly algorithm. The main novelty being the new short-read repeat resolution algorithm—RResolver. The new algorithm follows the low memory paradigm and efficiently uses short-read information to simplify the assembly graph.

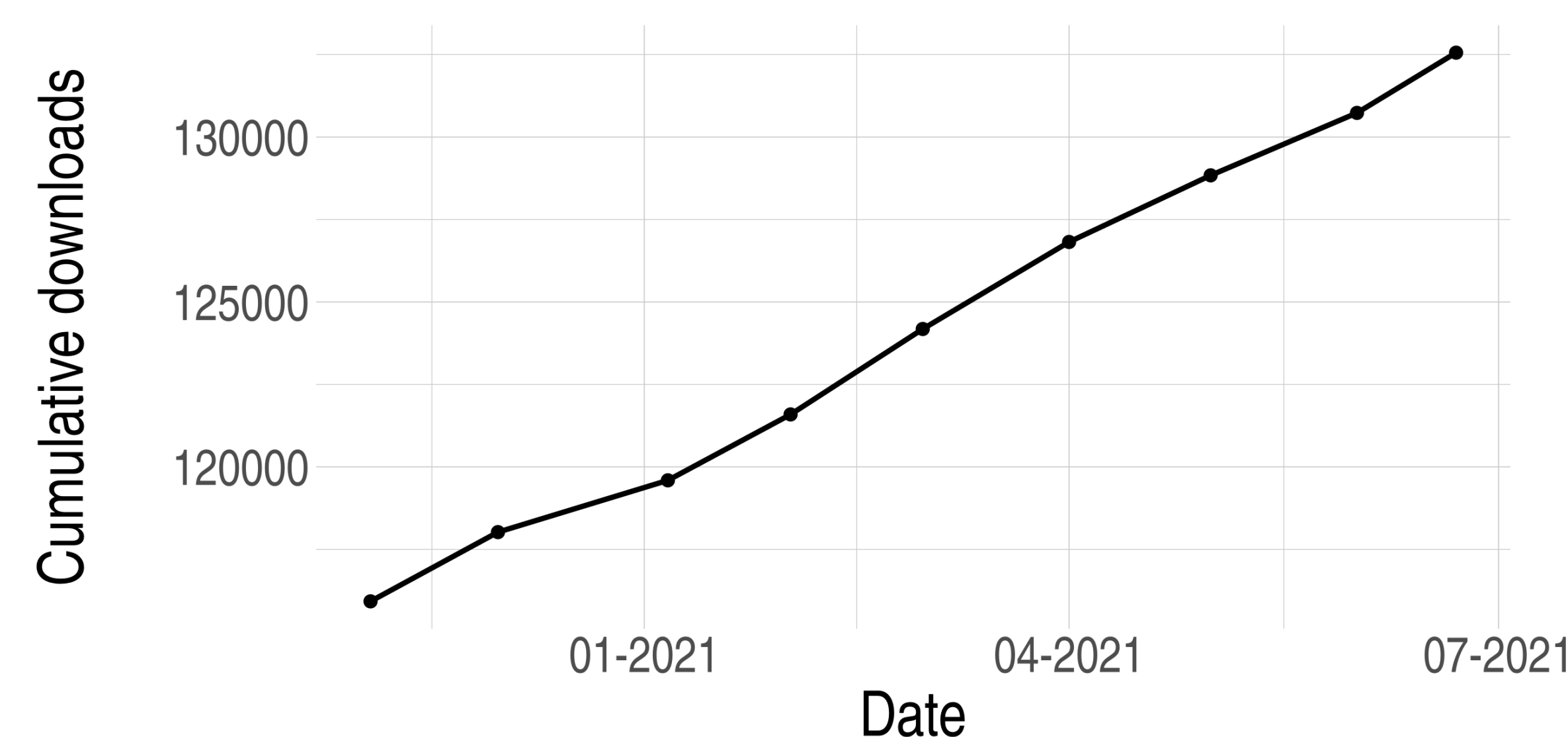


Assembly quality

- To demonstrate assembly quality improvement, comparison between ABySS 2.0 and ABySS 2.5 was made in NGA50 contiguity and complete BUSCOs recovered. *H. sapiens* 150bp paired-end reads from NA12878 and NA24631, and 250bp reads from NA24143 and NA24385 were assembled.

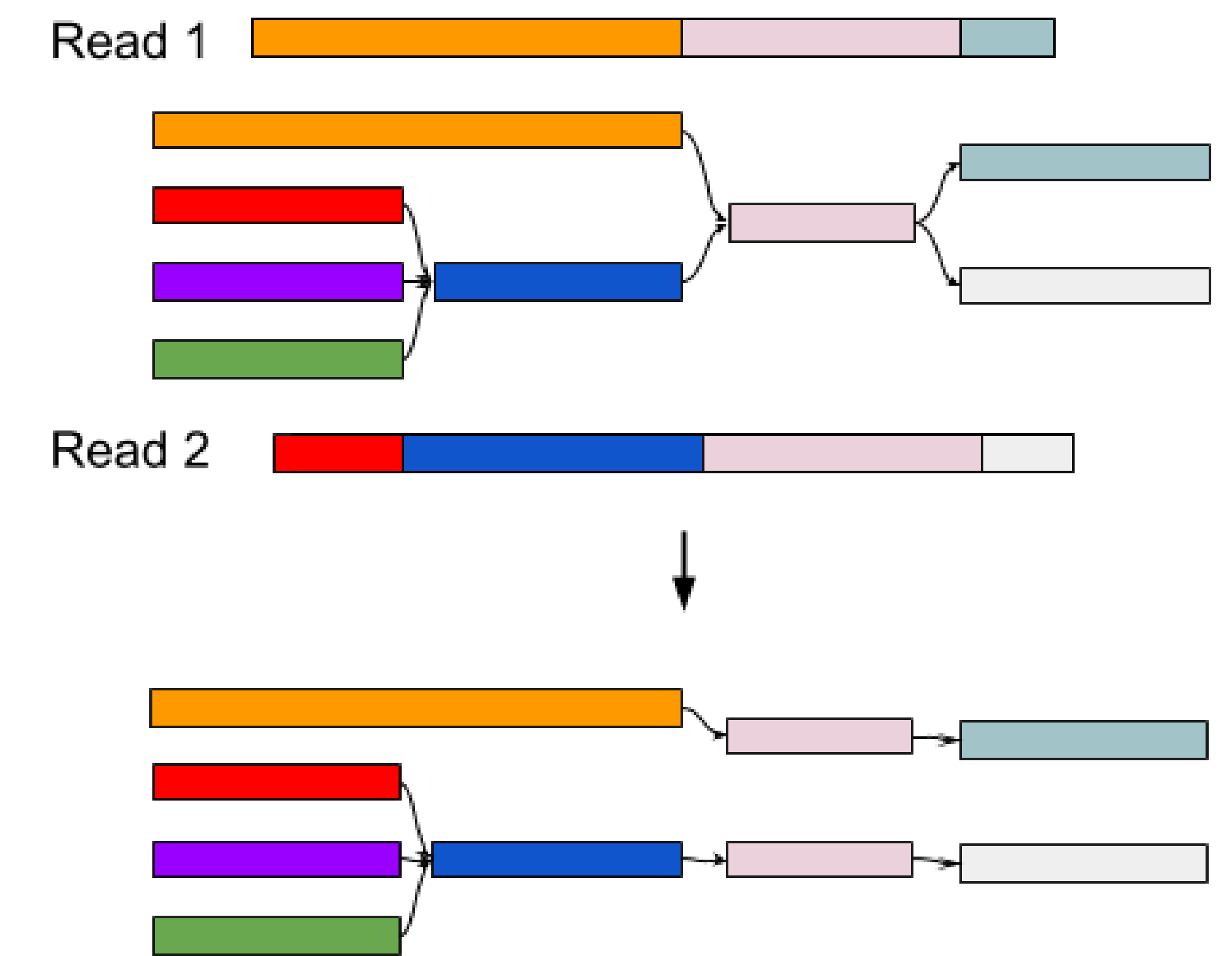


- Further, we show the download counts for the ABySS assembler over time, demonstrating the relevance of the tool more than a decade later.



Repeat resolution algorithm

- The RResolver algorithm utilizes the short-read range information to find paths in the assembly graph. The de Bruijn Graph node size is currently always shorter than the read size, and so some information is left unused, which the RResolver algorithm recovers.



- By identifying the presence of reads that span across the repeat, RResolver duplicates the repeat sequence where each instance is joined with corresponding graph nodes.
- The simplified graph allows for easier navigation and subsequently more correct decision making.

Conclusions

- From the initial release to ABySS 2.0, and now ABySS 2.5, the assembler has come a long way in delivering high quality *de novo* genome assemblies with low resource usage.
- The latest addition to the ABySS pipeline, the RResolver algorithm, further improves assembly quality, while keeping the resource usage low.

Funding

