bvSFM (bit-vector sampled FM-index) is a tool for sequence alignment. Specifically, it implements an exact search algorithm that counts the number of matches of arbitrary length reads on a reference genome. bvSFM indexes a genome with an FM Index (based on the Burrows-Wheeler Transform or BWT). FM Index is a compact data structure suitable for fast matches of short reads to large reference genomes. For the human genome, its memory footprint is typically around 3.2 gigabytes of RAM.

bvSFM uses an optimized FM-index data structure layout and codification that packs all relevant data needed in a query step within a single cache block, minimizing the memory bandwidth demand. bvSFM achieves best results when executed on multicore systems integrating high bandwidth memory, for instance an Intel Xeon Phi processor KNLa (codenamed Knights Landing, or KNL).

bvSFM is distributed under the GPLv3 license, and it runs on the command line under Linux.

If you use bvSFM for your published research, please cite the following paper:


bvSFM can be used as a CPU benchmark. It performs intensive integer computation. It can also be used to measure random memory access bandwidth.

Exact Matching Using FM-Index

Given a pattern Q[1..p], the FM-index allows to find all occurrences of Q in a reference text T.
The search process takes two steps: Count and Locate. The first step is a rank query process that calculates the number of occurrences of $Q$ in $T$. The second step uses the indexes of these rows to access the suffix array, where it finds the position of every occurrence of $Q$ in the text $T$. The suffix array is usually a very large compressed data structure. However, its size for the human genome is about 12 GB (3 gigabases x 4 bytes), so it can be stored without compression in modern systems. This way, the Locate step is very simple, it only requires an access to the suffix array. bvSFM only implements the Count step.

**k2d64bv version**

The Sampled FM-index (SFM) stores one set of counters out of every $d$ sets in the original Occ structure. This variant introduces a trade-off between memory footprint and computing cost.

The k-step FM-Index searches $k$ symbols in a query in a single step. This version reduces computing cost and improves data locality of the sampled version but increases slightly memory footprint.

This repository releases a k2d64bv version, that is, a sampling factor ($d$) of 64 and a k-step value of 2.