MUNI
FI

Datové struktury, algoritmy a nástroje pro zpracování genomických dat

IV110/IV114/E4014 Projekt z bioinformatiky (a systémové biologie)

MUNI FI

NGS sequencing

Read length
Gbp per run

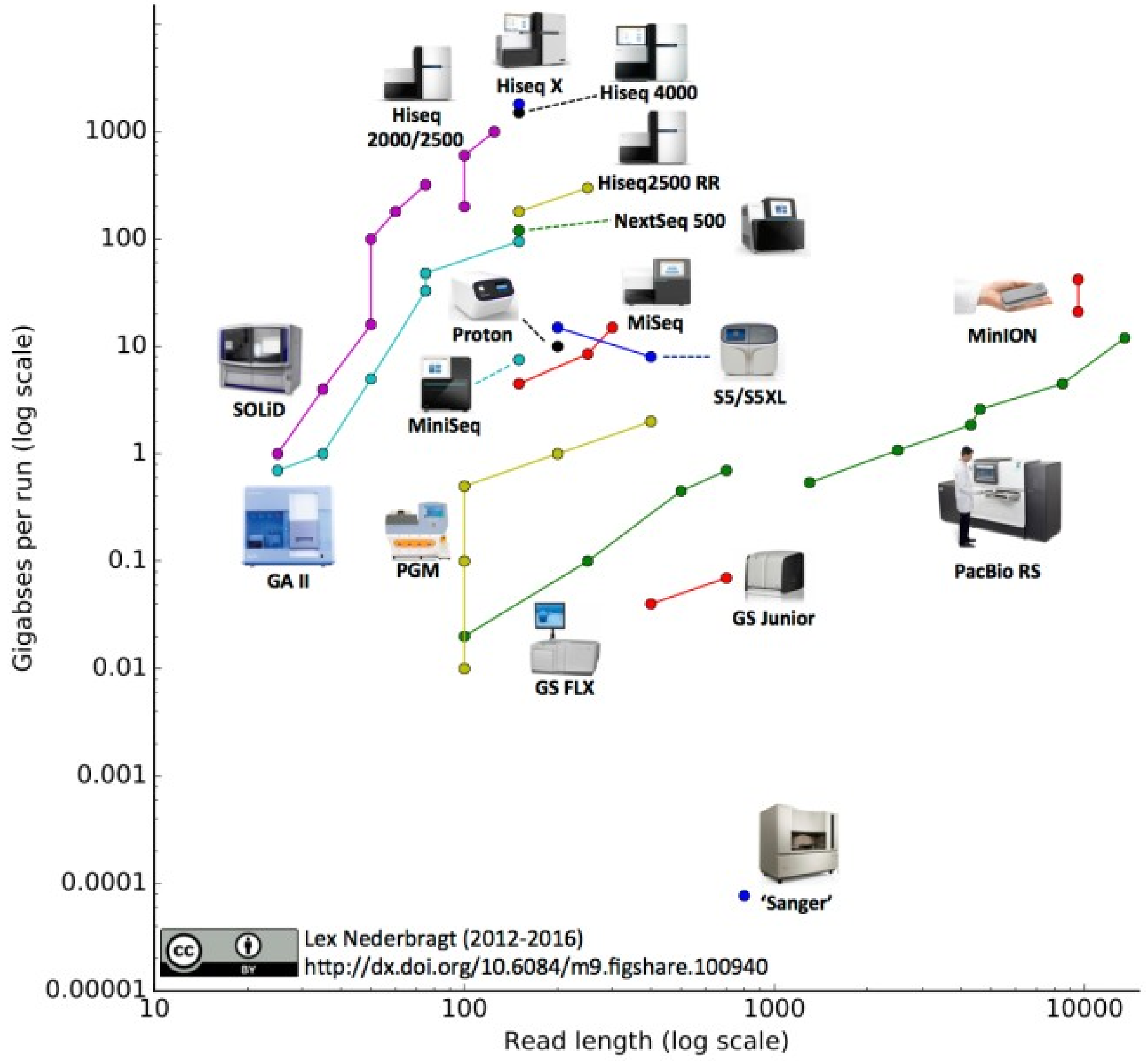
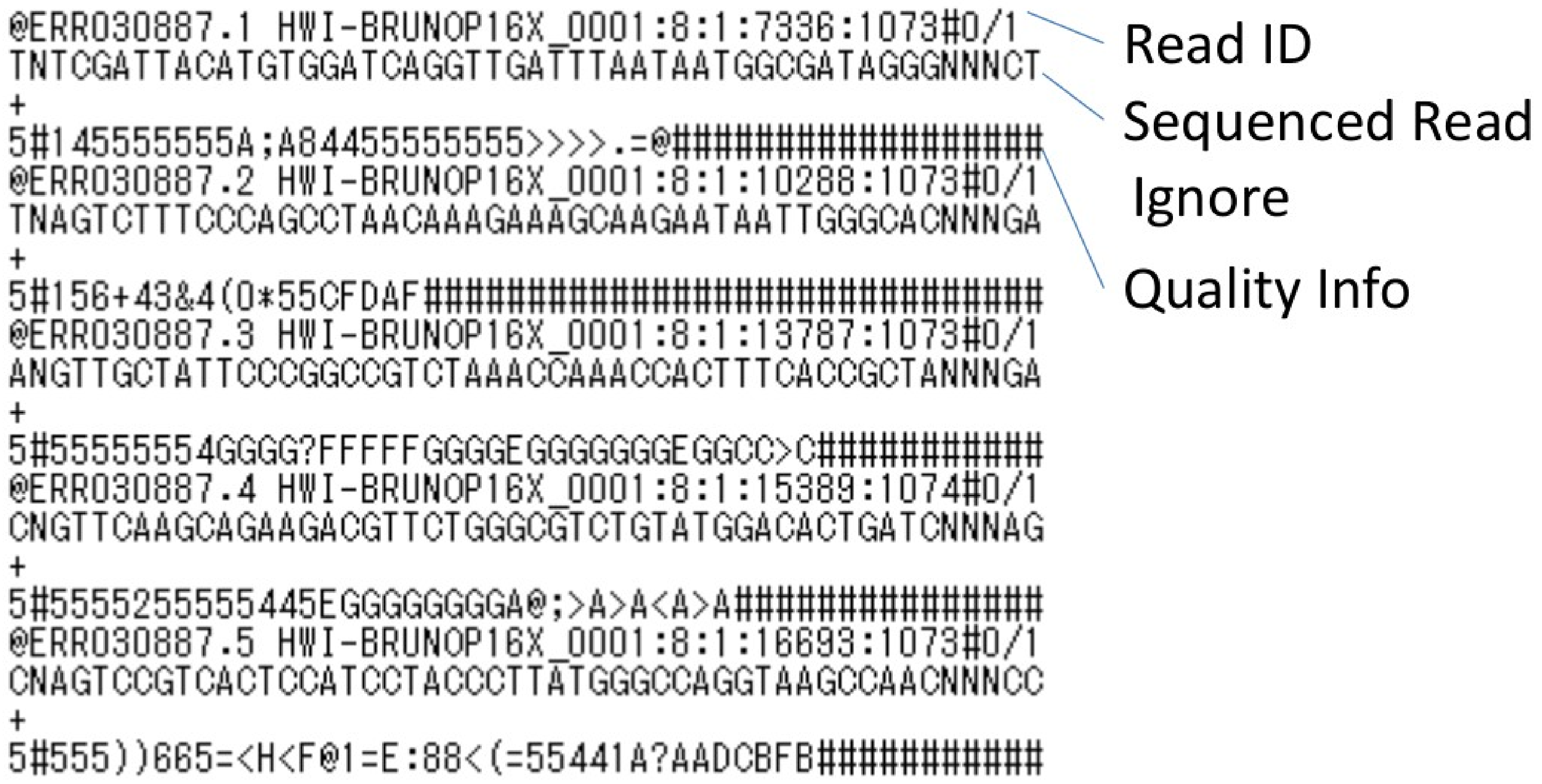


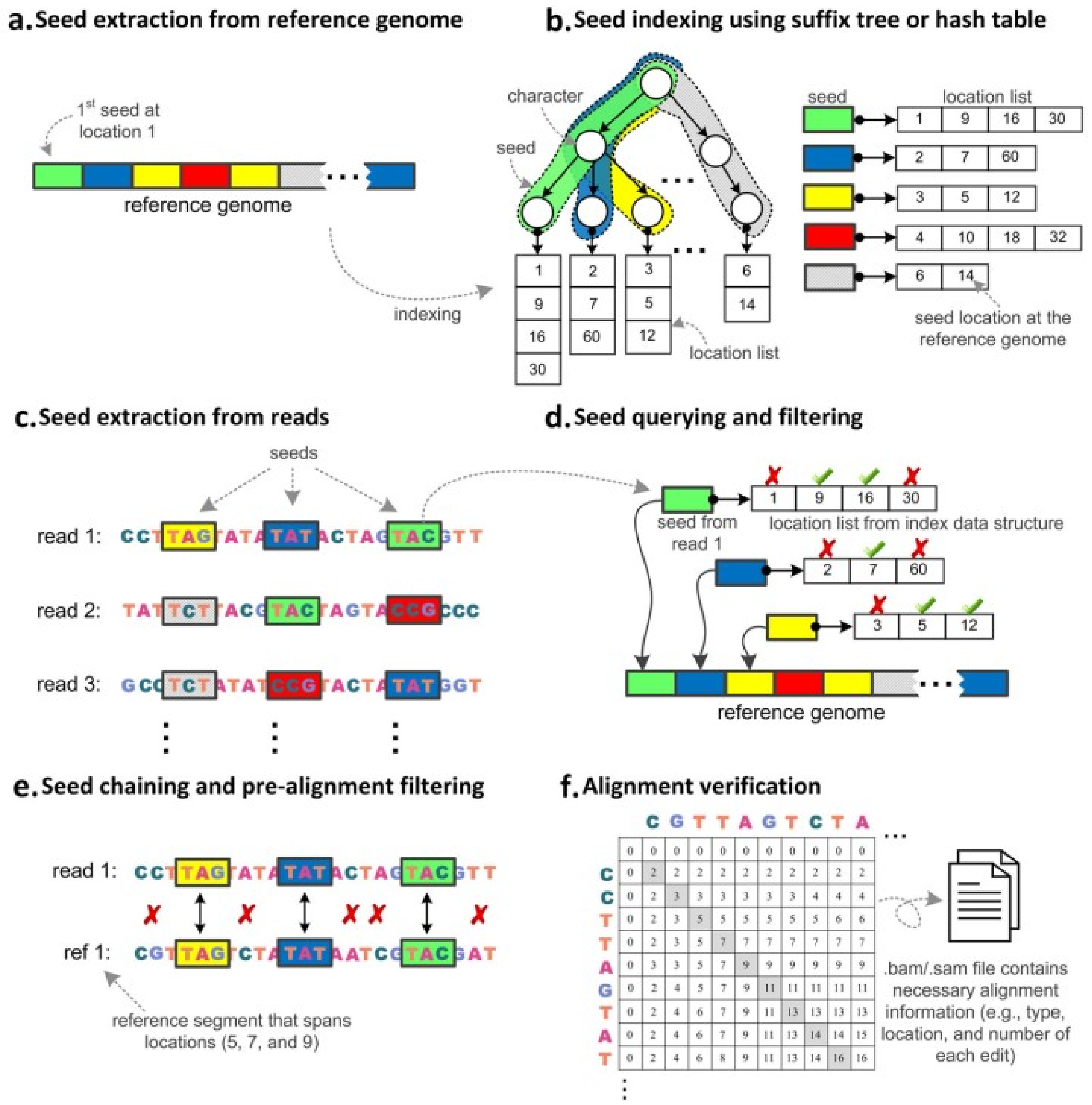
Figure 1.2: Comparison of sequencers based on their sequencing capacity and the length of the reads they produce (Nederbragt, 2016).

MUNI
FI

FASTQ data format


```
@ERR030887.1 HWI-BRUNOP16X_0001:8:1:7336:1073#0/1
TNTCGATTACATGTGGATCAGGTTGATTTAATAATGGCGATAGGGNNNCT
+
5#145555555A;A8445555555>>>>.=@###################
@ERR030887.2 HWI-BRUNOP16X_0001:8:1:10288:1073#0/1
TNAGTCTTTCCCAGCCTAACAAAGAAAGCAAGAATAATTGGGCACNNNGA
+
5#156+43&4(0*55CFDAF###############################
@ERR030887.3 HWI-BRUNOP16X_0001:8:1:13787:1073#0/1
ANGTTGCTATTCCCGGCCGTCTAAACCAAACCACTTTCACCGCTANNNGA
+
5#5555554GGGG?FFFFFGGGGEGGGGGGEGGCC>C#############
@ERR030887.4 HWI-BRUNOP16X_0001:8:1:15389:1074#0/1
CNGTTCAAGCAGAAGACGTTCTGGGCGTCTGTATGGACACTGATCNNNAG
+
5#5552555555445EGGGGGGGA@;>A>A<A>A################
@ERR030887.5 HWI-BRUNOP16X_0001:8:1:16693:1073#0/1
CNAGTCCGTCACTCCATCCTACCCTTATGGGCCAGGTAAGCCAACNNNCC
+
5#555))665=<H<F@1=E:88<(=55441A?AADCBFB###########
```

Efficient read mapping algorithms are based on k-mers



a. Seed extraction from reference genome
1st seed at location 1
reference genome
indexing
b. Seed indexing using suffix tree or hash table
character
seed
location list
seed
location list
seed location at the reference genome
c. Seed extraction from reads
seeds
read 1:
read 2:
read 3:
d. Seed querying and filtering
seed from read 1
location list from index data structure
reference genome
e. Seed chaining and pre-alignment filtering
read 1:
ref 1:
reference segment that spans locations (5, 7, and 9)
f. Alignment verification
.bam/.sam file contains necessary alignment information (e.g., type, location, and number of each edit)

MUNI
FI

Suffix trees

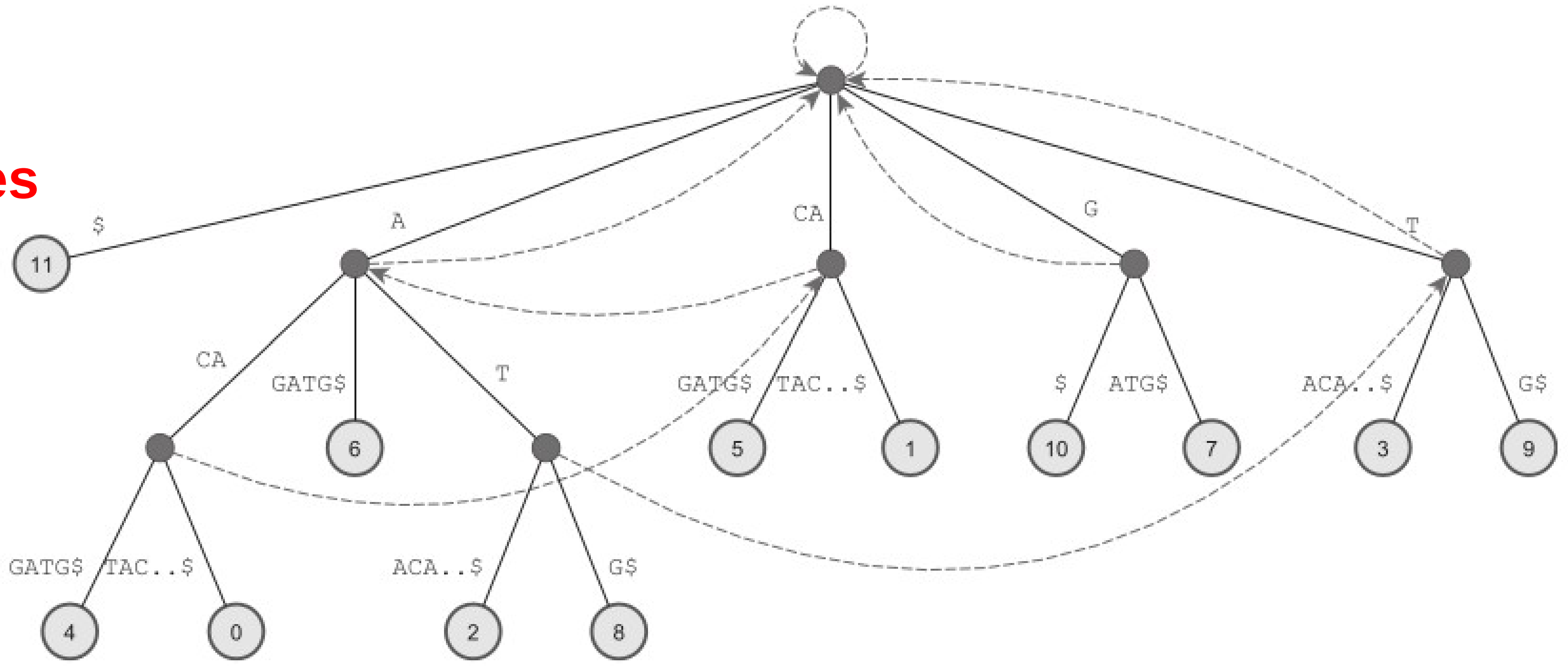


Figure 1. Suffix tree for string *S* = ACATACAGATG, where $ is the special end-character. Each number *i* inside a leaf represents suffix *S*[*i*..] of the string *S*. Dashed arrows correspond to suffix links. Edges are arranged in lexicographical order. For the sake of brevity, only the first characters followed by two dots and the special end-character $ are shown for edge labels that spell out the rest of the suffix corresponding to the leaf the edge is connected with.

BWT-based index

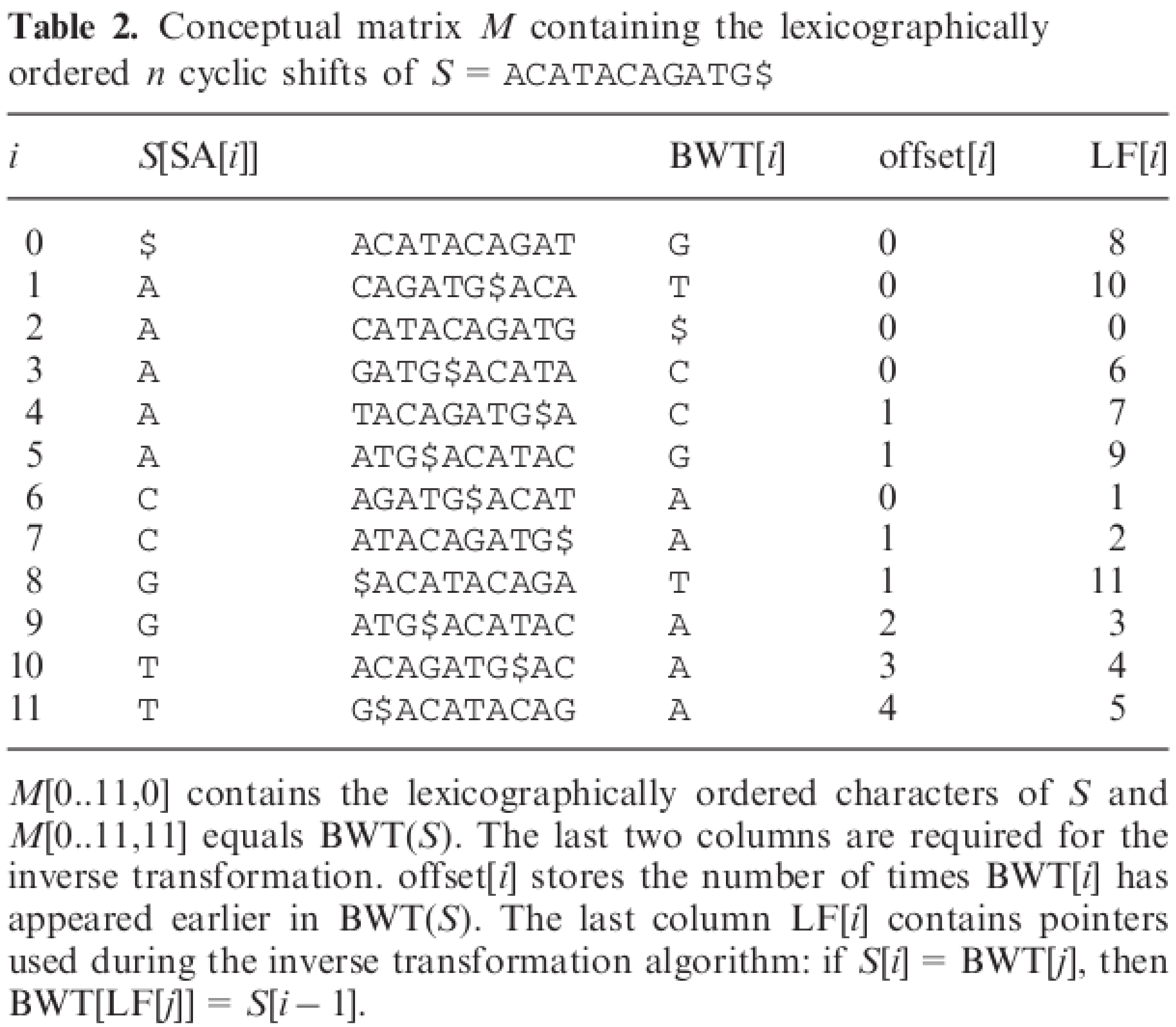


Table 2. Conceptual matrix M containing the lexicographically ordered n cyclic shifts of S = ACATACAGATG$

i	S[SA[i]]		BWT[i]	offset[i]	LF[i]
0	$	ACATACAGAT	G	0	8
1	A	CAGATG$ACA	T	0	10
2	A	CATACAGATG	$	0	0
3	A	GATG$ACATA	C	0	6
4	A	TACAGATG$A	C	1	7
5	A	ATG$ACATAC	G	1	9
6	C	AGATG$ACAT	A	0	1
7	C	ATACAGATG$	A	1	2
8	G	$ACATACAGA	T	1	11
9	G	ATG$ACATAC	A	2	3
10	T	ACAGATG$AC	A	3	4
11	T	G$ACATACAG	A	4	5

M[0..11,0] contains the lexicographically ordered characters of S and M[0..11,11] equals BWT(S). The last two columns are required for the inverse transformation. offset[i] stores the number of times BWT[i] has appeared earlier in BWT(S). The last column LF[i] contains pointers used during the inverse transformation algorithm: if $S[i]$ = BWT[j], then BWT[LF[j]] = $S[i-1]$.

Enhanced suffix array and BWT-based indexes

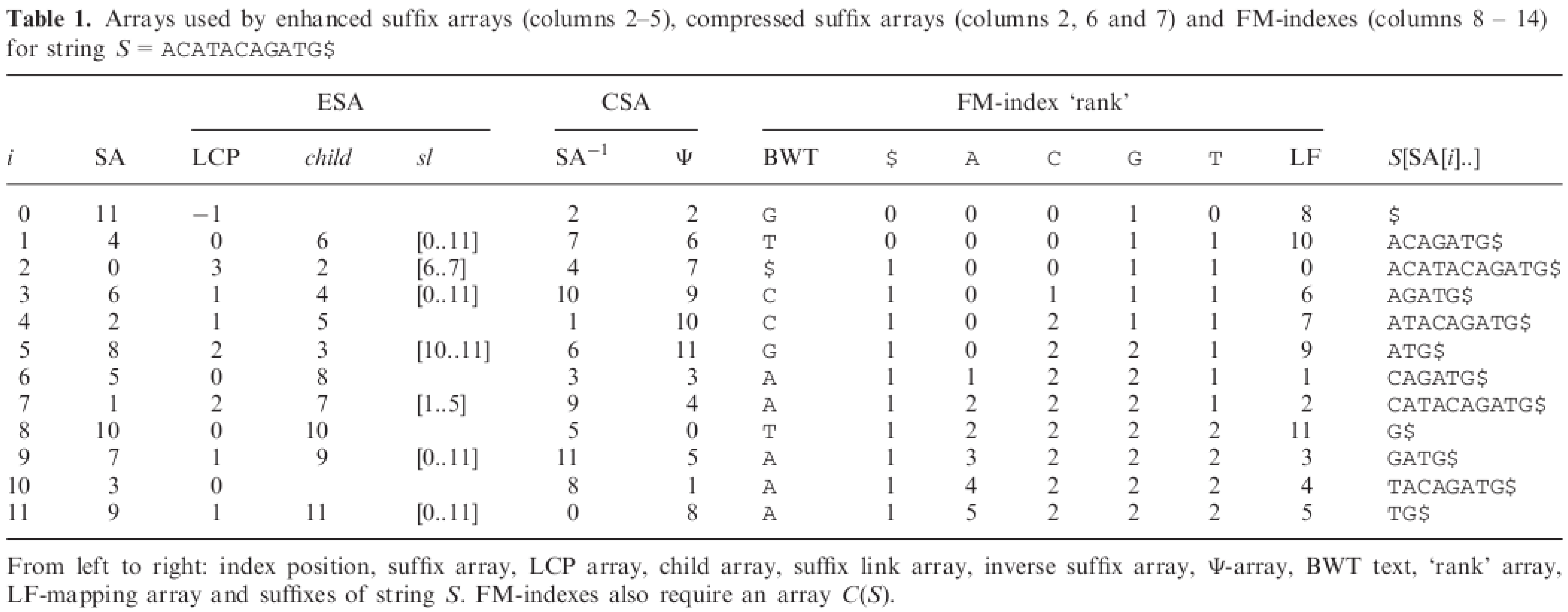
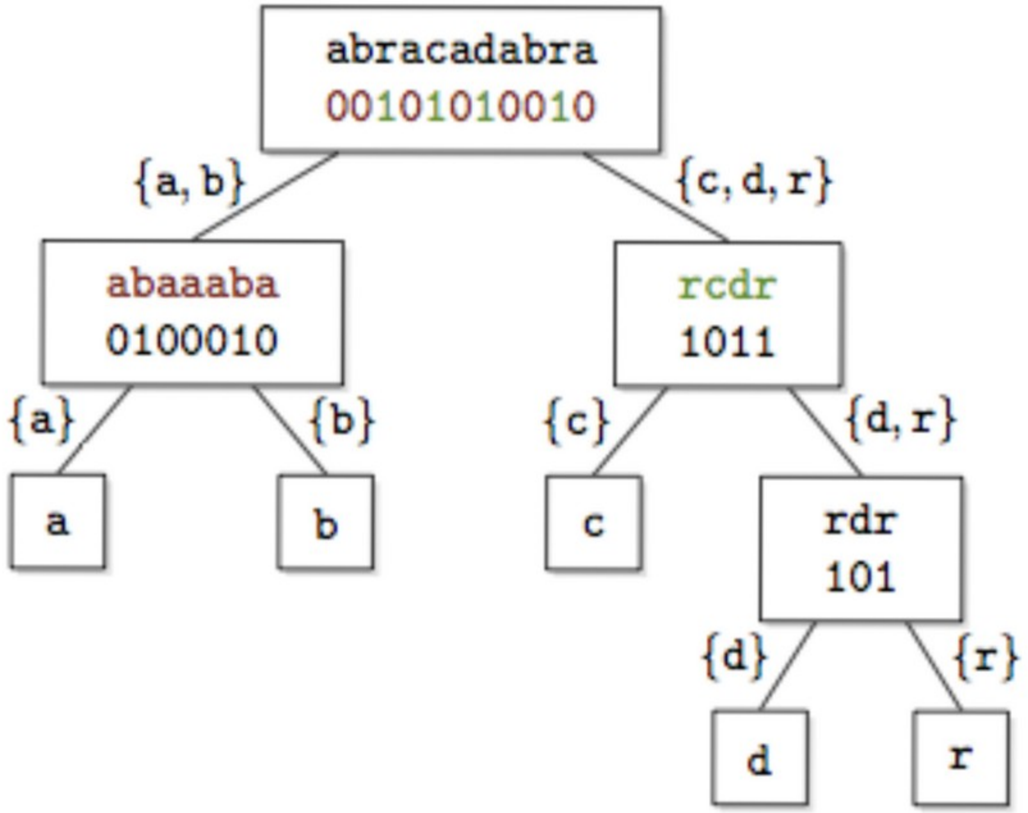


Table 1. Arrays used by enhanced suffix arrays (columns 2–5), compressed suffix arrays (columns 2, 6 and 7) and FM-indexes (columns 8 – 14) for string S = ACATACAGATG$

		ESA			CSA		FM-index ‘rank’							
i	SA	LCP	*child*	*sl*	SA^{-1}	Ψ	BWT	$	A	C	G	T	LF	*S*[SA[*i*]..]
0	11	−1			2	2	G	0	0	0	1	0	8	$
1	4	0	6	[0..11]	7	6	T	0	0	0	1	1	10	ACAGATG$
2	0	3	2	[6..7]	4	7	$	1	0	0	1	1	0	ACATACAGATG$
3	6	1	4	[0..11]	10	9	C	1	0	1	1	1	6	AGATG$
4	2	1	5		1	10	C	1	0	2	1	1	7	ATACAGATG$
5	8	2	3	[10..11]	6	11	G	1	0	2	2	1	9	ATG$
6	5	0	8		3	3	A	1	1	2	2	1	1	CAGATG$
7	1	2	7	[1..5]	9	4	A	1	2	2	2	1	2	CATACAGATG$
8	10	0	10		5	0	T	1	2	2	2	2	11	G$
9	7	1	9	[0..11]	11	5	A	1	3	2	2	2	3	GATG$
10	3	0			8	1	A	1	4	2	2	2	4	TACAGATG$
11	9	1	11	[0..11]	0	8	A	1	5	2	2	2	5	TG$

From left to right: index position, suffix array, LCP array, child array, suffix link array, inverse suffix array, Ψ-array, BWT text, ‘rank’ array, LF-mapping array and suffixes of string *S*. FM-indexes also require an array *C*(*S*).

Wavelet tree



Read mapping tool
Performance
- CPU
- RAM

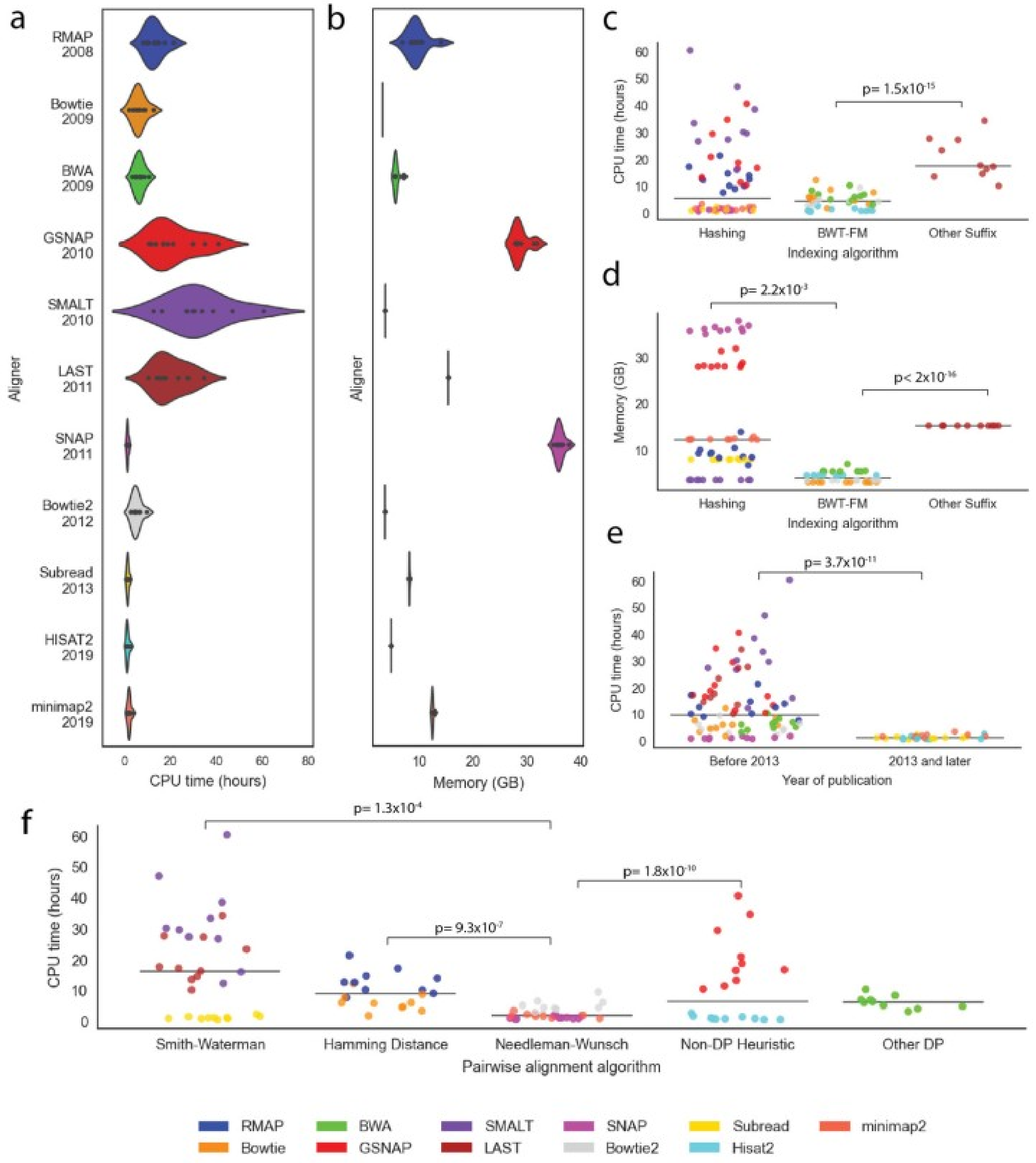
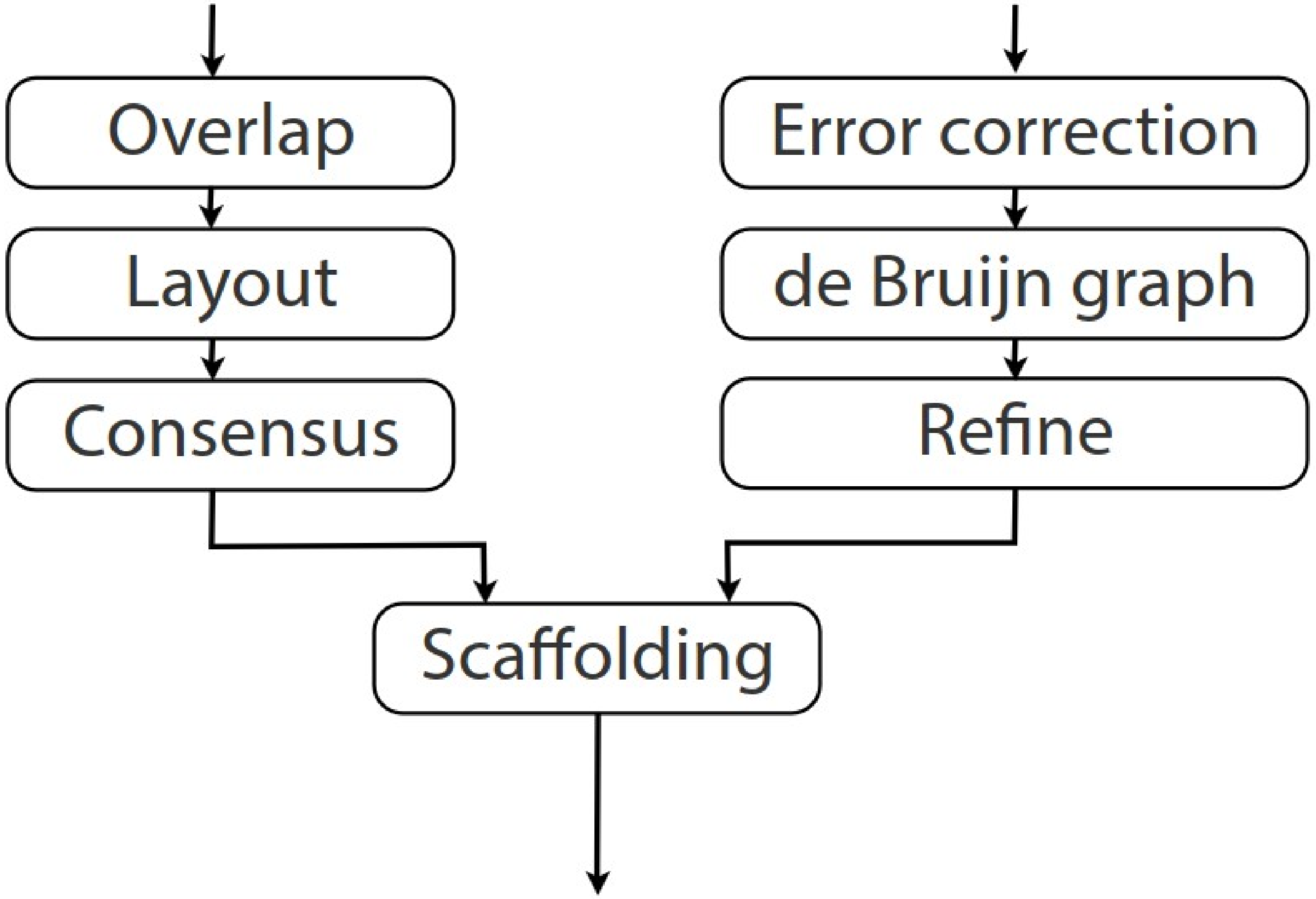


Fig. 4 The effect of read alignment algorithms on the speed of alignment and computational resources.

Assembly alternatives



Overlap graph for overlap-layout-consensus assembly

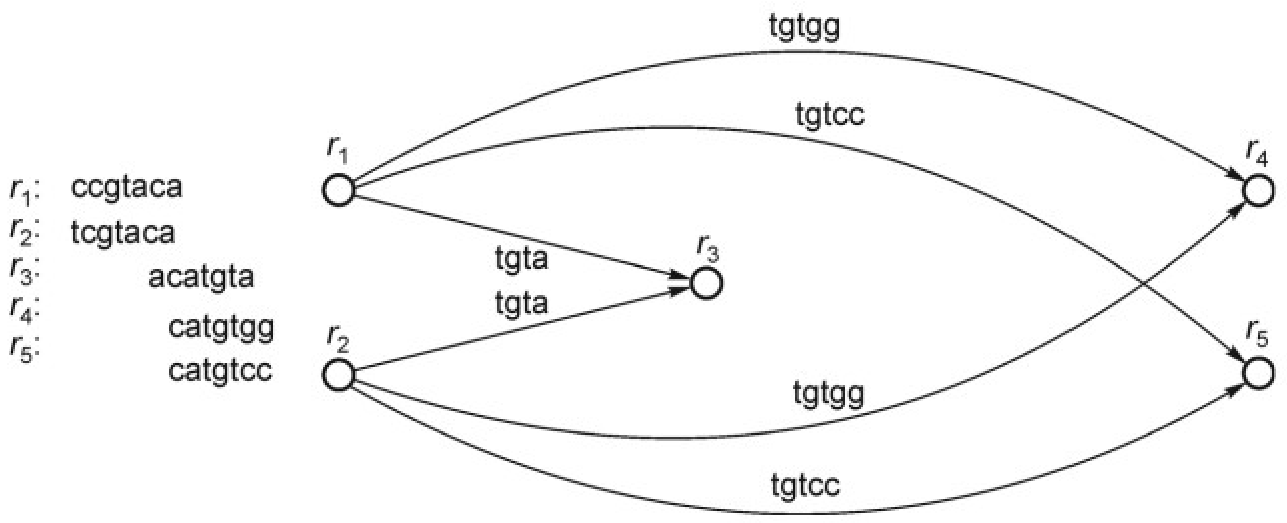


Figure 2. Example of the overlap graph for five reads r_1, r_2, r_3, r_4, r_5. Each edge (r_i, r_j) is labelled by the *extension* $e_{i,j}$.

DeBruijn graph for assembly

1: ccgt
2: tcgt
3: acat
4: catg
5: cgta
6: gtac
7: taca
8: atgt
9: tgtg
10: gtgg
11: tgta
12: tgtc
13: gtcc

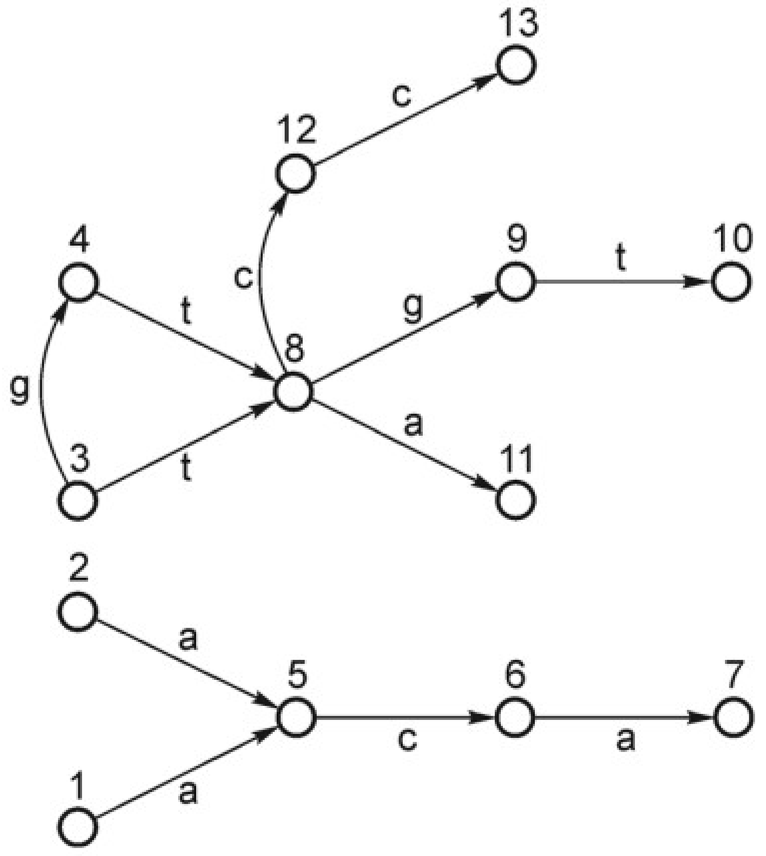


Figure 3. Example of the ***de Bruijn*** **graph for *k* = 3 of the two reads *ccgtac* and *catgtg*.** The nodes are the sixteen *k*-mers reported on the left. Each arc is labelled by the last character of its second node.

Tools for NGS
Read processing

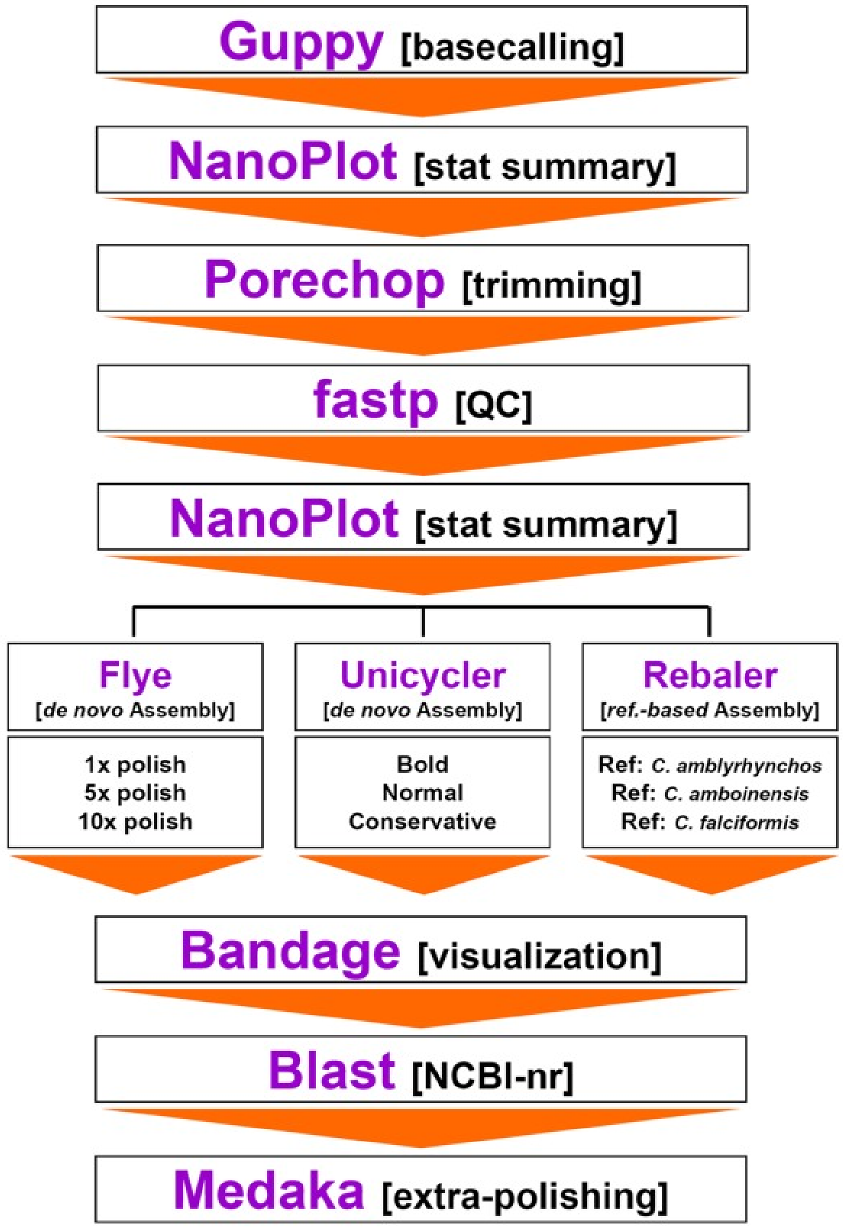
MAPPING

- Bowtie2, STAR, BWA-MEM

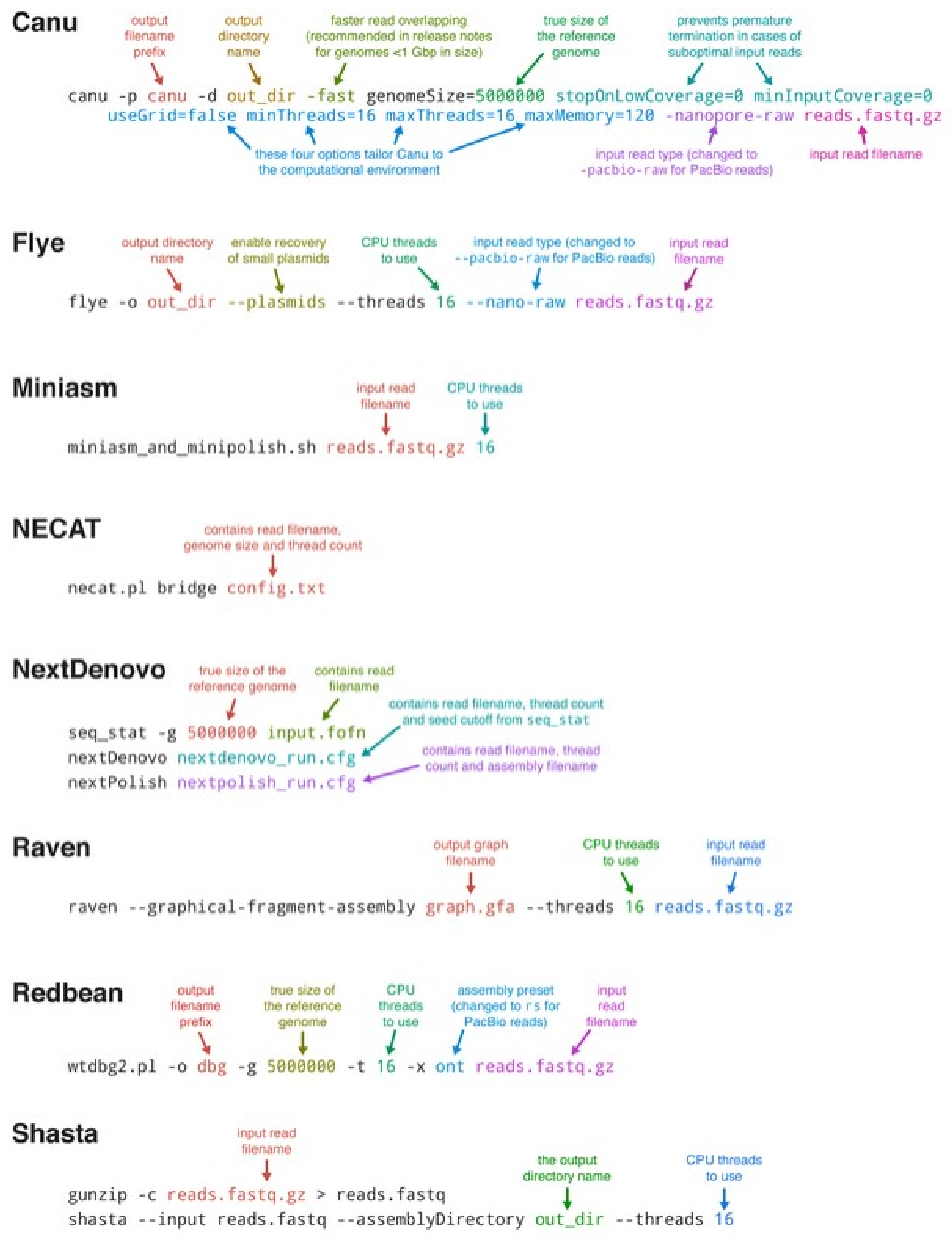
ASSEMBLY

- SHORT READ
  - Velvet, AbySS, SOAPdenovo
- LONG READ
  - Flye, Canu, miniasm, Minipolish, NECAT, NextDeNovo, Nextpolish, Raven, Redbean, Shasta
- HYBRID
  - SPADes, MaSuRCA, Unicycler

Nanopore read Processing pipeline



Command-line examples for various nanopore assembly tools



Canu

output filename prefix
output directory name
faster read overlapping (recommended in release notes for genomes <1 Gbp in size)
true size of the reference genome
prevents premature termination in cases of suboptimal input reads

```
canu -p canu -d out_dir -fast genomeSize=5000000 stopOnLowCoverage=0 minInputCoverage=0
    useGrid=false minThreads=16 maxThreads=16 maxMemory=120 -nanopore-raw reads.fastq.gz
```

these four options tailor Canu to the computational environment
input read type (changed to -pacbio-raw for PacBio reads)
input read filename

Flye

output directory name
enable recovery of small plasmids
CPU threads to use
input read type (changed to --pacbio-raw for PacBio reads)
input read filename

```
flye -o out_dir --plasmids --threads 16 --nano-raw reads.fastq.gz
```

Miniasm

input read filename
CPU threads to use

```
miniasm_and_minipolish.sh reads.fastq.gz 16
```

NECAT

contains read filename, genome size and thread count

```
necat.pl bridge config.txt
```

NextDenovo

true size of the reference genome
contains read filename
contains read filename, thread count and seed cutoff from seq_stat
contains read filename, thread count and assembly filename

```
seq_stat -g 5000000 input.fofn
nextDenovo nextdenovo_run.cfg
nextPolish nextpolish_run.cfg
```

Raven

output graph filename
CPU threads to use
input read filename

```
raven --graphical-fragment-assembly graph.gfa --threads 16 reads.fastq.gz
```

Redbean

output filename prefix
true size of the reference genome
CPU threads to use
assembly preset (changed to rs for PacBio reads)
input read filename

```
wtdbg2.pl -o dbg -g 5000000 -t 16 -x ont reads.fastq.gz
```

Shasta

input read filename
the output directory name
CPU threads to use

```
gunzip -c reads.fastq.gz > reads.fastq
shasta --input reads.fastq --assemblyDirectory out_dir --threads 16
```

Genome assembly quality assessment

Contig N50

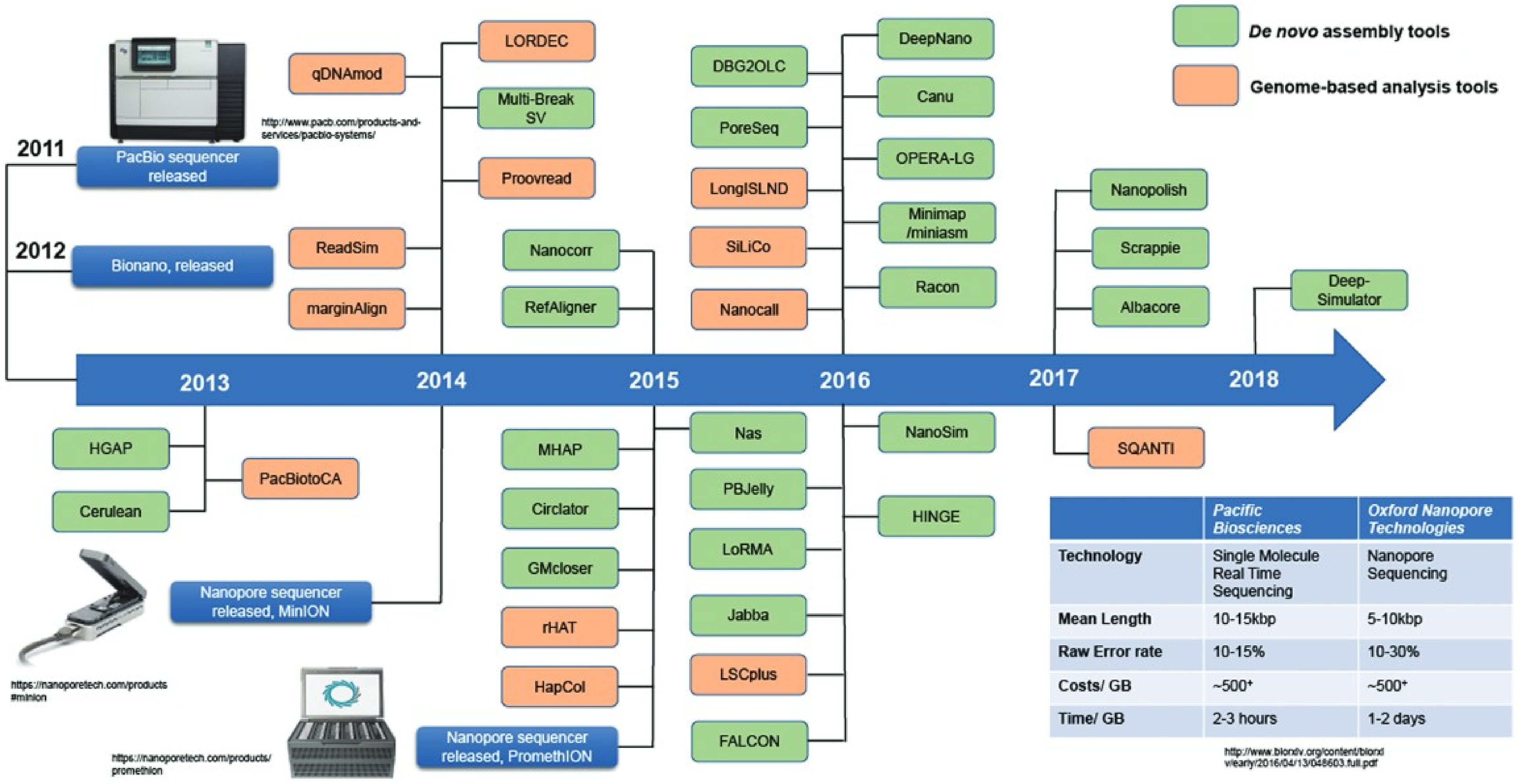
COMPASS

BUSCO score

LAI score

MUNI
FI

3rd gen-seq tools



	Pacific Biosciences	*Oxford Nanopore Technologies*
Technology	Single Molecule Real Time Sequencing	Nanopore Sequencing
Mean Length	10-15kbp	5-10kbp
Raw Error rate	10-15%	10-30%
Costs/ GB	~500*	~500*
Time/ GB	2-3 hours	1-2 days

Figure 1. Milestones in TGS analysis software development. The green box refers to the *de novo* assembly tool while the orange box refers to the genome-based analysis tool.

MUNI FI

JBrowse

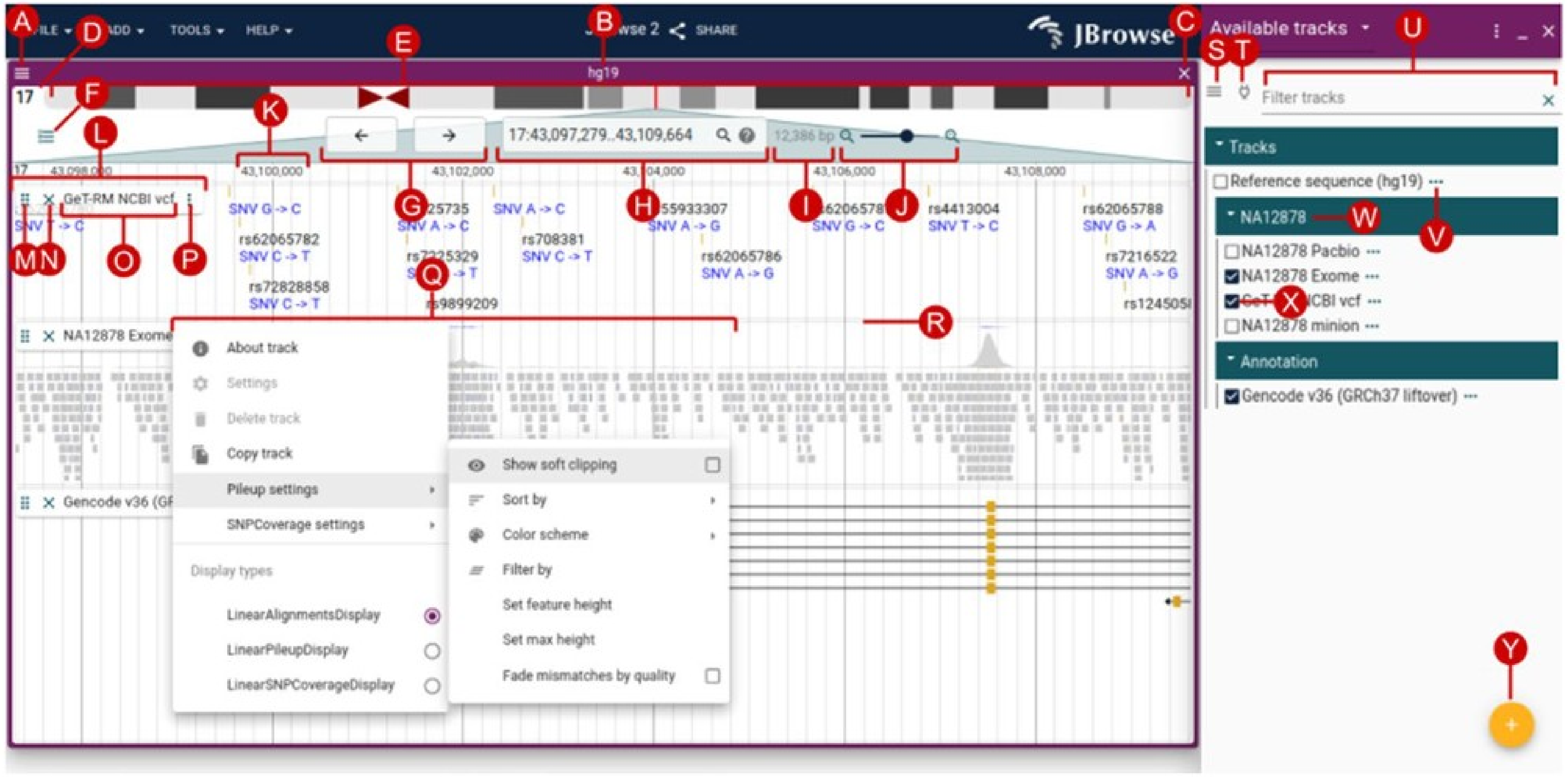


Fig. 2 The Linear Genome View is the core view of JBrowse, allowing flexible and interactive examination

MUNI
FI

JBrowse

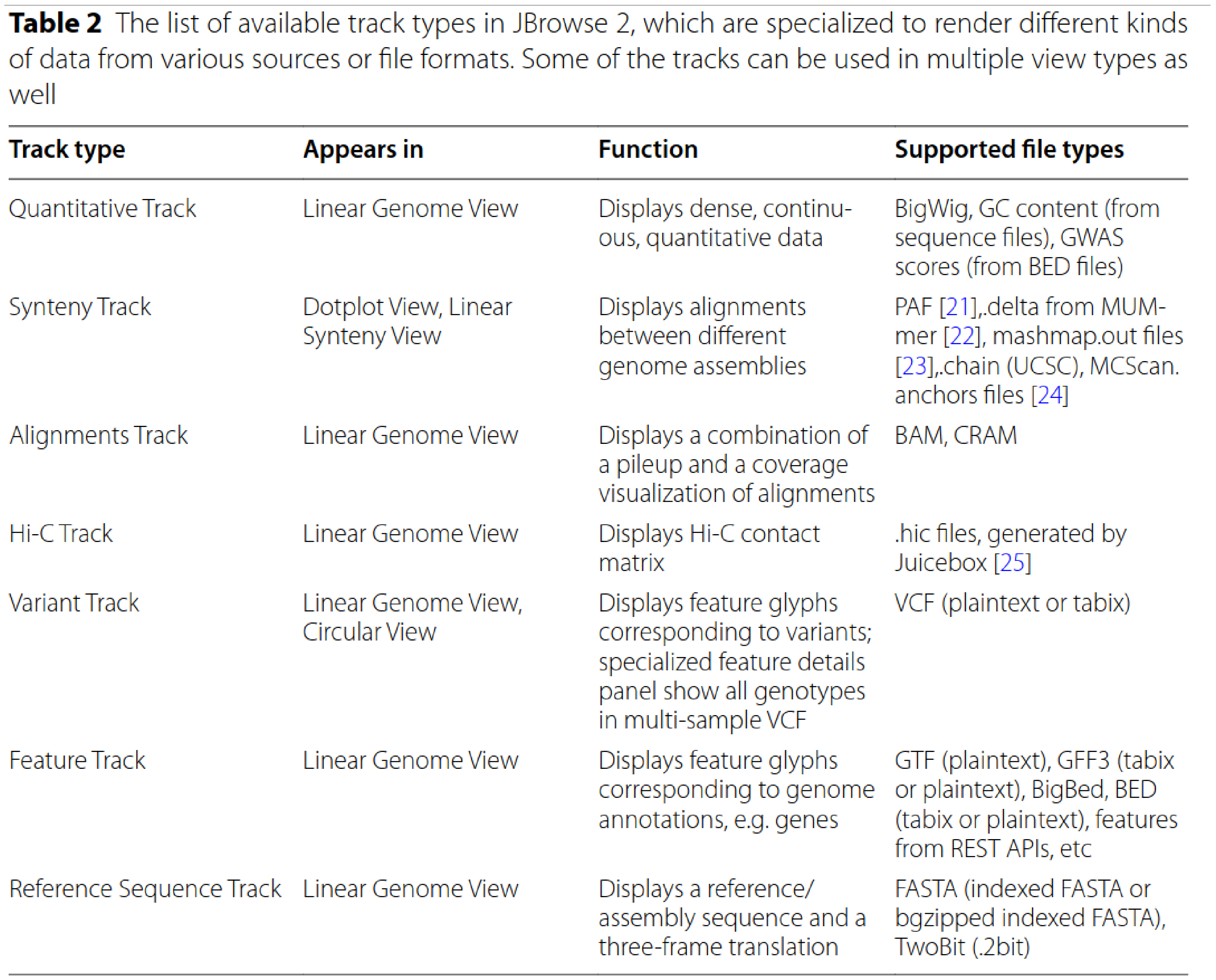
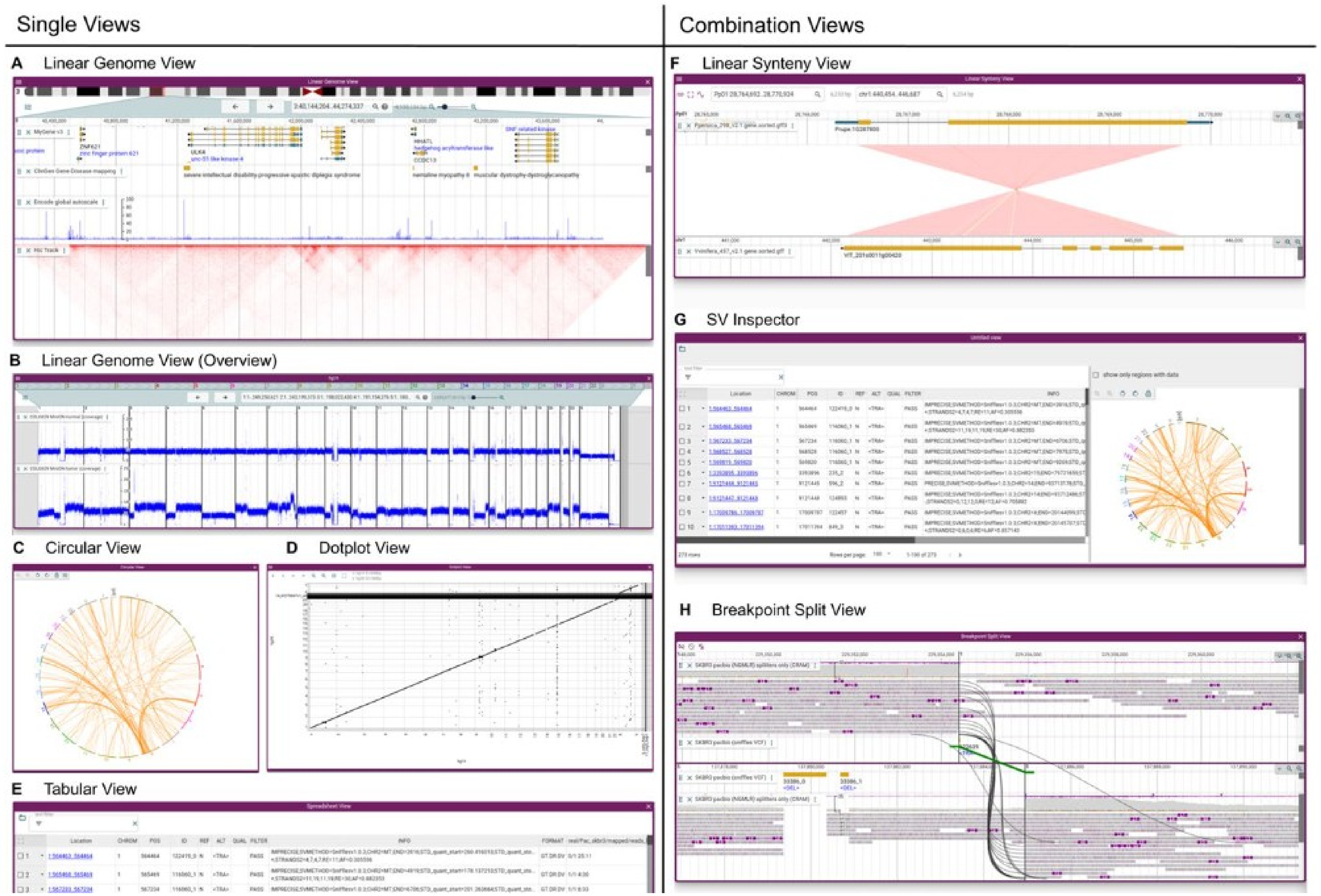


Table 2 The list of available track types in JBrowse 2, which are specialized to render different kinds of data from various sources or file formats. Some of the tracks can be used in multiple view types as well

Track type	Appears in	Function	Supported file types
Quantitative Track	Linear Genome View	Displays dense, continuous, quantitative data	BigWig, GC content (from sequence files), GWAS scores (from BED files)
Synteny Track	Dotplot View, Linear Synteny View	Displays alignments between different genome assemblies	PAF [21],.delta from MUMmer [22], mashmap.out files [23],.chain (UCSC), MCScan.anchors files [24]
Alignments Track	Linear Genome View	Displays a combination of a pileup and a coverage visualization of alignments	BAM, CRAM
Hi-C Track	Linear Genome View	Displays Hi-C contact matrix	.hic files, generated by Juicebox [25]
Variant Track	Linear Genome View, Circular View	Displays feature glyphs corresponding to variants; specialized feature details panel show all genotypes in multi-sample VCF	VCF (plaintext or tabix)
Feature Track	Linear Genome View	Displays feature glyphs corresponding to genome annotations, e.g. genes	GTF (plaintext), GFF3 (tabix or plaintext), BigBed, BED (tabix or plaintext), features from REST APIs, etc
Reference Sequence Track	Linear Genome View	Displays a reference/assembly sequence and a three-frame translation	FASTA (indexed FASTA or bgzipped indexed FASTA), TwoBit (.2bit)

MUNI
FI

JBrowse



Single Views
Combination Views
A Linear Genome View
B Linear Genome View (Overview)
C Circular View
D Dotplot View
E Tabular View
F Linear Synteny View
G SV Inspector
H Breakpoint Split View