cgatools Methods

Software v1.1.0

Table of Contents

Preface

The Complete Genomics Analysis Tools (**cgatools**) is an open source project to provide tools for downstream analysis of Complete Genomics data. This document describes the motivation and design decisions for **cgatools**.

Conventions

This document uses the following notational conventions:

Notation	Description
italic	A field name from a data file. For example, the *varType* field in the variations data file indicates the type of variation identified between the assembled genome and the reference genome. Also used to indicate the values found in data files. For example, *ChromosomeName* indicates that the value found in the data file is the name of a given chromosome.
bold_italic	A file name from the data package. For example, each package contains the file ***manifest.all***.

CGI Data

Complete Genomics, Inc. (CGI) delivers complete genome sequencing data to customers and collaborators. The data include sequence reads, their mappings to a reference human genome, and variations detected against the reference human genome. This document describes tools developed to analyze this data.

References

- Assembly Pipeline Release Notes — Indicates new features and enhancements by release.
- *Complete Genomics Variation FAQ* — Answers to frequently asked questions about Complete Genomics variation data.
- *Complete Genomics Technology Whitepaper* — A concise description of the Complete Genomics sequencing technology, including the library construction process and the ligation-based assay approach. It is available in the "Resources" section of the Complete Genomics website. [www.completegenomics.com]
- *Getting Started with CGI's Data FAQ* — Answers to questions about preparing to receive the hard drives of data.
- Complete Genomics *Data File Formats* — A description of the organization and content of the format for complete genome sequencing data delivered by Complete Genomics. It is available on the Complete Genomics website. [www.completegenomics.com]
- Complete Genomics *Science* Article — An article from the Complete Genomics chief scientific officer describing the process of how CGI maps reads (*Science* 327 (5961), 78. [DOI: 10.1126/science.1181498]). We also recommend you read the *Complete Genomics Service FAQ* as background for this document. [www.drmanac.com]
- SAM— The Sequence Alignment/Map format is a generic format for storing large nucleotide sequence alignments. This **cgatools** description is based on the SAM Format Specification "Sequence Alignment/Map (SAM) Format," Version 0.1.2-draft (August 20, 2009). [http://samtools.sourceforge.net/SAM1.pdf]

Reference Tools

At times, **cgatools** uses the reference sequence in a random-access manner. The most common reference sequence format, FASTA, is not ideal for processing tasks that require random access because the entire sequence must be read into memory at the start of the program, and this memory cannot be shared among processes.

CRR File Format

cgatools uses its own file format, Compact Randomly Accessible Reference (CRR), to represent a reference sequence. The CRR file format stores two bits per base of reference, plus lookup tables to resolve regions of the reference that are represented by ambiguous IUPAC codes. CRR files are memory mapped, so that processes can share a reference, and the overall memory requirement due to the reference for all processes is less than 1 GB. The CRR file format does not preserve character case—FASTA reference files often use case to denote the region's repeat status—and considers all the bases described in the reference FASTA sequence as upper case.

FASTA Reference Sequences

Complete Genomics supports two references. The first, which we refer to as "build 36," consists of the assembled nuclear chromosomes from NCBI build 36 (not unplaced or alternate loci) plus Yoruban mitochondrion NC_001807.4. This assembly is also known as UCSC hg18. The second reference, which we refer to as "build 37," consists of the assembled nuclear chromosomes from GRCh37 (not unplaced or alternate loci), plus the Cambridge Reference Sequence for the mitochondrion (NC_012920.1). This assembly (though with an alternate mitochondrial sequence) is also known as UCSC hg19. The resulting FASTA sequences are available here:

ftp://ftp.completegenomics.com/ReferenceFiles/build36.fa.bz2

ftp://ftp.completegenomics.com/ReferenceFiles/build37.fa.bz2

fasta2ccr

This tool converts FASTA sequences into a single reference CRR file.

crr2fasta

This tool converts CRR sequence files to the FASTA file format.

decodecrr

This command retrieves the sequence for a given range of a chromosome.

listcrr

This command lists the chromosomes, contigs, or regions of ambiguous sequence within the reference, depending on the parameters. The contigs described by listcrr are defined to be the contiguous sequence bases separated by at least `min-contig-gap-length`, no-call bases, where `min-contig-gap-length` defaults to 50. The default contigs correspond to the notion of contig employed in the Complete Genomics data, such as reference scores.

After you have successfully created a build 36 or 37 CRR file (that is, converted the downloaded reference into CRR using fasta2crr) for use with Complete Genomics data, the listcrr command returns the output shown in Figure 1:

Figure 1: listcrr Output for Build 36

```
ChromosomeId Chromosome    Length Circular Md5
           0       chr1 247249719    false 9ebc6df9496613f373e73396d5b3b6b6
           1       chr2 242951149    false b12c7373e3882120332983be99aeb18d
           2       chr3 199501827    false 0e48ed7f305877f66e6fd4addbae2b9a
           3       chr4 191273063    false cf37020337904229dca8401907b626c2
           4       chr5 180857866    false 031c851664e31b2c17337fd6f9004858
           5       chr6 170899992    false bfe8005c536131276d448ead33f1b583
           6       chr7 158821424    false 74239c5ceee3b28f0038123d958114cb
           7       chr8 146274826    false 1eb00fe1ce26ce6701d2cd75c35b5ccb
           8       chr9 140273252    false ea244473e525dde0393d353ef94f974b
           9      chr10 135374737    false 4ca41bf2d7d33578d2cd7ee9411e1533
          10      chr11 134452384    false 425ba5eb6c95b60bafbf2874493a56c3
          11      chr12 132349534    false d17d70060c56b4578fa570117bf19716
          12      chr13 114142980    false c4f3084a20380a373bbbdb9ae30da587
          13      chr14 106368585    false c1ff5d44683831e9c7c1db23f93fbb45
          14      chr15 100338915    false 5cd9622c459fe0a276b27f6ac06116d8
          15      chr16  88827254    false 3e81884229e8dc6b7f258169ec8da246
          16      chr17  78774742    false 2a5c95ed99c5298bb107f313c7044588
          17      chr18  76117153    false 3d11df432bcdc1407835d5ef2ce62634
          18      chr19  63811651    false 2f1a59077cfad51df907ac25723bff28
          19      chr20  62435964    false f126cdf8a6e0c7f379d618ff66beb2da
          20      chr21  46944323    false f1b74b7f9f4cdbaeb6832ee86cb426c6
          21      chr22  49691432    false 2041e6a0c914b48dd537922cca63acb8
          22       chrX 154913754    false d7e626c80ad172a4d7c95aadb94d9040
          23       chrY  57772954    false 62f69d0e82a12af74bad85e2e4a8bd91
          24       chrM     16571     true d2ed829b8a1628d16cbeee88e88e39eb
```

After you have successfully created a build 37 CRR file (that is, converted the downloaded reference into CRR using fasta2crr) for use with Complete Genomics data, the listcrr command returns the output shown in Figure 2:

Figure 2: listcrr Output for Build 37

```
ChromosomeId Chromosome    Length Circular Md5
           0       chr1 249250621    false 1b22b98cdeb4a9304cb5d48026a85128
           1       chr2 243199373    false a0d9851da00400dec1098a9255ac712e
           2       chr3 198022430    false 641e4338fa8d52a5b781bd2a2c08d3c3
           3       chr4 191154276    false 23dccd106897542ad87d2765d28a19a1
           4       chr5 180915260    false 0740173db9ffd264d728f32784845cd7
           5       chr6 171115067    false 1d3a93a248d92a729ee764823acbbc6b
           6       chr7 159138663    false 618366e953d6aaad97dbe4777c29375e
           7       chr8 146364022    false 96f514a9929e410c6651697bded59aec
           8       chr9 141213431    false 3e273117f15e0a400f01055d9f393768
           9      chr10 135534747    false 988c28e000e84c26d552359af1ea2e1d
          10      chr11 135006516    false 98c59049a2df285c76ffb1c6db8f8b96
          11      chr12 133851895    false 51851ac0e1a115847ad36449b0015864
          12      chr13 115169878    false 283f8d7892baa81b510a015719ca7b0b
          13      chr14 107349540    false 98f3cae32b2a2e9524bc19813927542e
          14      chr15 102531392    false e5645a794a8238215b2cd77acb95a078
          15      chr16  90354753    false fc9b1a7b42b97a864f56b348b06095e6
          16      chr17  81195210    false 351f64d4f4f9ddd45b35336ad97aa6de
          17      chr18  78077248    false b15d4b2d29dde9d3e4f93d1d0f2cbc9c
          18      chr19  59128983    false 1aacd71f30db8e561810913e0b72636d
          19      chr20  63025520    false 0dec9660ec1efaaf33281c0d5ea2560f
          20      chr21  48129895    false 2979a6085bfe28e3ad6f552f361ed74d
          21      chr22  51304566    false a718acaa6135fdca8357d5bfe94211dd
          22       chrX 155270560    false 7e0e2e580297b7764e31dbc80c2540dd
          23       chrY  59373566    false 1e86411d73e6f00a10590f976be01623
          24       chrM     16569     true c68f52674c9fb33aef52dcf399755519
```

Genome Comparison Tools

A Note on Conventions

To call low certainty regions or "no-call" regions, Complete Genomics augments the alphabet {A, C, G, T} with two additional characters:

- The "N" character corresponds to a one-base sequence that may be any of {A, C, G, T}.
- The "?" character corresponds to zero or more bases of unknown sequence.

The Problem of Genome Comparison

Genome comparison is the problem of identifying genomic sequence that is identical, compatible (perhaps with no-calls), or incompatible, with sequence from another genome. Within the space of genome comparison problems, there are three common tasks:

1. Is a genome identical, compatible, or incompatible with the reference genome at a given location?
2. Is a genome identical, compatible, or incompatible with a known common sequence?
3. Is a genome identical, compatible, or incompatible with a particular genome at a given location within the reference genome?

The particular way a genome is described by re-sequencing technologies goes a long way towards solving genome comparison problems 1 and 2: genomes are represented as a set of differences (or variants) against the reference genome. The Complete Genomics variant file format differs from most other common variant file formats in that in addition to describing the variants, it also distinguishes regions of the genome that are called as reference from those that are no-called. As we will see later, this distinction is essential in solving many comparison problems.

Let's see how the Complete Genomics variant file would describe the following situation, where chr1 is a diploid chromosome and chr2 is a haploid chromosome:

chr1 reference:	CATGACCCGCAAA-TCTGAAACTATCTGGCCCTTGGCAGGGG--A
chr1 haplotype 1:	?ATGACCTGCAAAATCTGAAACT--CTGGCCCTTGGCAGGGGGGA
chr1 haplotype 2:	?ATGACCCGCAAAATCTGAAACTATCTGGCTNTTGGCAGGGT--A
chr2 reference:	TGATATTTTTCATCAACATTACAGGCA
chr2:	TGATATTTTTNATCAACACGACAGGCA

Figure 3 shows the corresponding variant file.

Figure 3: Variant File

>locus	ploidy	allele	chromosome	begin	end	varType	reference	alleleSeq	totalScore	hapLink	xRef
1	2	all	chr1	0	1	no-call	=	?			
2	2	all	chr1	1	7	ref	=	=			
3	2	1	chr1	7	8	snp	C	T	87	1	dbsnp:123
3	2	2	chr1	7	8	ref	C	C	58	2	dbsnp:123
4	2	all	chr1	8	13	ref	=	=			
5	2	1	chr1	13	13	ins		A	36		
5	2	2	chr1	13	13	ins		A	42		
6	2	all	chr1	13	22	ref	=	=			
7	2	1	chr1	22	24	del	AT		47	1	
7	2	2	chr1	22	24	ref	AT	AT	55	2	
8	2	all	chr1	24	29	ref	=	=			
9	2	1	chr1	29	31	ref	CC	CC	57	1	
9	2	2	chr1	29	31	no-call-ri	CC	TN	65	2	
10	2	all	chr1	31	40	ref	=	=			
11	2	1	chr1	40	41	ref	G	G	101	1	
11	2	1	chr1	41	41	ins		GG	120	1	
11	2	2	chr1	40	41	snp	G	T	479	2	
12	2	all	chr1	41	42	ref	=	=			
13	1	all	chr2	0	10	ref	=	=			
14	1	1	chr2	10	11	no-call-rc	C	N	47		
15	1	all	chr2	11	18	ref	=	=			
16	1	1	chr2	18	20	sub	TT	CG	102		
17	1	all	chr2	20	27	ref	=	=			

The genome is first aligned to the reference, and then split into loci. Each locus may describe multiple alleles (if ploidy > 1), and for each allele at each locus, there may be one or more lines (or "calls") to describe the sequence. The variant file describes 0-based offsets within the reference chromosome.

In the variant file in Figure 2, locus 3 describes a heterozygous SNP (one-base polymorphism on one allele, reference on the other allele). Locus 5 describes a homozygous insertion (in which the confidence is slightly higher for allele 2 than allele 1). The allele column is used to distinguish the alleles of calls within a locus. For example, the "ref" and "ins" calls of locus 11 are on the same allele, whereas the "snp" call is on the opposite allele. To declare that two calls of different loci are on the same haplotype, the format uses the *hapLink* field. Calls known to be on the same haplotype have the same *hapLink* value; calls with different *hapLink* values may or may not be on the same haplotype (the phasing is no-called).

For a detailed reference of the Complete Genomics variant file format, see the *Data File Format* document provided with your genome (See "References").

Problems Not Solved by Variant File Format

One problem you may have noticed is that the problem of aligning a genome to the reference is not necessarily well-defined. For example in Figure 2, the homozygous insertion at locus 5 could have also been described by the same homozygous insertion three bases to the left. Or the substitution at locus 16 could have been described as two SNPs. Comparing two genomes that describe the same sequence in different ways can be complicated.

We could make canonicalization rules such as "always use the rightmost insertion for any insertion that has multiple possible representations" or "always decompose an allele consisting of a SNP, two reference bases, then another SNP, into separate calls." Indeed, Complete Genomics has rules like these that are generally

followed. But there are at least three remaining problems in solving the genome comparison problems described above:

- Known variants are not always described in their canonical form.

  For example, entries rs34330821 and rs34544546 in the dbSNP database of known variants describe equivalent insertions that are 18 bases apart. This may seem superficial, in that dbSNP entries that are not described in their canonical form can be canonicalized. But if our canonical form uses less decomposition than the dbSNP submission, this may not be possible; if a dbSNP submission has been decomposed, the submission has lost information about nearby variants that exist on the same haplotype.

- Canonical forms of near-identical sequences are not necessarily near-identical.

  For example, suppose we have a genome that is equivalent to a SNP and an insert against the reference, as described in canonicalization 1:

  ```
  Reference:          TG A TGTGAATTGGTG ---------------------------- AGT
  Canonicalization 1: TG C TGTGAATTGGTG TAGTGTGAATGAGTGTGTGAATTGGTG AGT

  Reference:          TG A--------------------------- TGTGAATTGGTGAGT
  Canonicalization 2: TG CTGTGAATTGGTGTAGTGTGAATGAGTG TGTGAATTGGTGAGT
  ```

  The insert in canonicalization 1 might be the simplest way to describe the genome if the SNP did not exist. But one could argue that the single substitution in canonicalization 2 is the simplest canonicalization of the genome, given that the SNP does exist. (This would be the case for a canonicalization which favors fewer calls.) It is not obvious by visual inspection that the insert from canonicalization 1 and the substitution of canonicalization 2 differ by only a SNP.

- No-calls may not be canonicalized like insertions or deletions, such that an insert may be compatible with another genome only when viewing a larger sequence of the genome.

  For example, suppose we have the following reference and the following genome:

  ```
  Reference:  CGAAAAAAA-TTTTCG
  Genome:     CGAAAAAAAATTTTCG
  ```

  Now suppose the genome reconstruction process discovers that an insertion has occurred, but it does not know if the first base in the run of A's is really an A, or perhaps was a C. In this case, we are forced to align the no-call at the beginning as follows:

  ```
  Reference:  CG-AAAAAAATTTTCG
  Genome:     CGNAAAAAAATTTTCG
  ```

  Length no-calls ("?") may further complicate the situation so that the alignment is unclear. For example, suppose in the same example above, in addition to not knowing if the first base of the run is an A or a C, we also don't know the length of the run of A's at all. Suppose also that we know that the run of T's has increased in length from four to five. There could be at least two reasonable alignments of the result, corresponding to a called insert or a called SNP:

  ```
  Reference:   CGAAAAAAATTTT-CG
  Alignment 1: CG?AAAAAATTTTTCG
  Reference:   CG-AAAAAAATTTTCG
  Alignment 2: CG?AAAAAATTTTTCG
  ```


Genome Comparison with cgatools

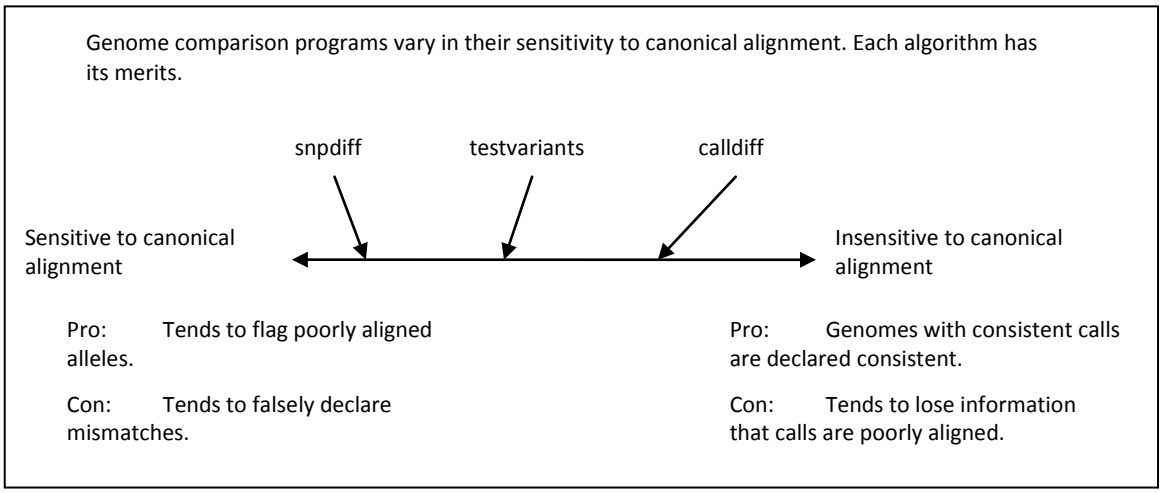
There is a wide spectrum of useful genome comparison methods, which range in their sensitivity to the canonical alignment of called sequence. Algorithms that are very sensitive to canonical alignment tend to declare sequences inconsistent when in fact they are consistent. Algorithms that are less sensitive to canonical alignment tend to be less discriminating in terms of the quality of the alignment of called sequence.

cgatools includes two genome comparison utilities that provide varying degrees of sensitivity to inconsistent canonical alignments:

- snpdiff can be used to compare the results of a SNP caller to a Complete Genomics variant file. It is quite sensitive to the canonical alignment of called sequence.
- calldiff can be used to compare two variant files. It is less sensitive to the canonical alignment of called sequence.

Figure 4 illustrates the tradeoffs between the two utilities.

Figure 4: Sensitivity of Genome Comparison Algorithms



snpdiff

The snpdiff tool compares SNP calls to a Complete Genomics variant file. It is particularly useful for comparing a Complete Genomics variant file to SNP calls provided by an alternative sequencing or genotyping platform that only produces SNP calls. The input SNP calls must be in a tab-delimited file with columns as in Figure 5. Note that the order of columns can vary but column titles must be conserved; the *Genotypes* column is optional.

Figure 5: SNP Calls As Input to snpdiff

```
Chromosome    Offset0Based  GenotypesStrand     Genotypes
chr13         17919222      +                   CC
chr13         17919650      +                   AT
chr13         17920392      +                   NN
chr13         17921548      +                   TT
```

Here, the *Genotypes* column specifies the base call for each allele. The output produced for this input may be something along the lines of Figure 6:

Figure 6: Output from snpdiff

Chromosome	Offset0Based	GenotypesStrand	Genotypes	Reference	Variants	DiscordantAlleles	NoCallAlleles
chr13	17919222	+	CC	T	CC	0	0
chr13	17919650	+	AT	A	AA	1	0
chr13	17920392	+	NN	G	GN	0	1
chr13	17921548	+	TT	T	.-	0	0

The result for each allele described in the *Variants* column above are any base call ("A", "C", "G", or "T"), a no-call ("N"), a deletion ("-"), or a larger variation that is not consistent with a SNP at all ("."). To compare the SNP calls to the calls in the variant file, snpdiff first determines the variant file's calls at the given position. The algorithm used is sensitive to the canonical alignment, and it is aggressive in terms of making a base call at positions where the call does not have a *varType* of "snp" or "ref". That being said, it is tested to be largely concordant with SNP calls made by several alternative technologies. A discordance found by snpdiff is likely to be a true discrepancy between the calls made by the SNP caller and the variant file. The algorithm employed by snpdiff is as follows, for each allele:

- Find the call in the variant file that overlaps the position in question. Use this call alone to determine the base call for the position in question.
- Walk the *alleleSeq* column of the call from the right and left until reaching the position in question. For each direction, any of the following outcomes may be reached:
  - WALK_OK – The position in question was reached.
  - WALK_EOS – The end of alleleSeq was reached before getting to the position in question.
  - WALK_INCOMPATIBLE – A base call incompatible with the reference base was found at some position before reaching the position in question.
  - WALK_LENGTH_NOCALL – A length no-call (represented by "?") is discovered before reaching the position in question.
- Combine the results of the walk from the right and left to determine the result. The results are combined by the following rules:
  - If the walk from the left and right both end up at the position of interest (WALK_OK):
    - If the base calls discovered by the two walks are in conflict, declare a larger variation (".").
    - If the base calls discovered by the two walks are consistent and at least one is called, use the base call.
    - If both walks end up with a no-call ("N"), the result is no-call.
  - If only one walk ends up at the position of interest (WALK_OK), use the base discovered by that walk.
  - If neither walk ends up at the position of interest, then:
    - If either walk ends up as WALK_LENGTH_NOCALL, mark the position as no-call ("N").
    - If either walk ends up as WALK_EOS, mark the position as deleted ("-").
    - Otherwise, mark the position as a larger variant (".").

Figure 7 shows some examples. The reference base we wish to determine a call for is highlighted in red:

Figure 7: Example Results from snpdiff

reference	alleleSeq	Walk L->R	Walk R->L	Outcome
A	C	WALK_OK: C	WALK_OK: C	C
ACGTACGT	ACGTACGT	WALK_OK: T	WALK_OK: T	T
G	CC	WALK_OK: C	WALK_OK: C	C
G	CG	WALK_OK: C	WALK_OK: G	.
ACGT	AGGN	WALK_OK: G	WALK_OK: G	G
ACGT	AGG?	WALK_OK: G	WALK_LENGTH_NOCALL	G
ACGT	?GG?	WALK_LENGTH_NOCALL	WALK_LENGTH_NOCALL	N
ACGT	CGGT	WALK_INCOMPATIBLE	WALK_OK: G	G
ACGT	CGGG	WALK_INCOMPATIBLE	WALK_INCOMPATIBLE	.
CACACAC	CAC	WALK_EOS	WALK_EOS	-

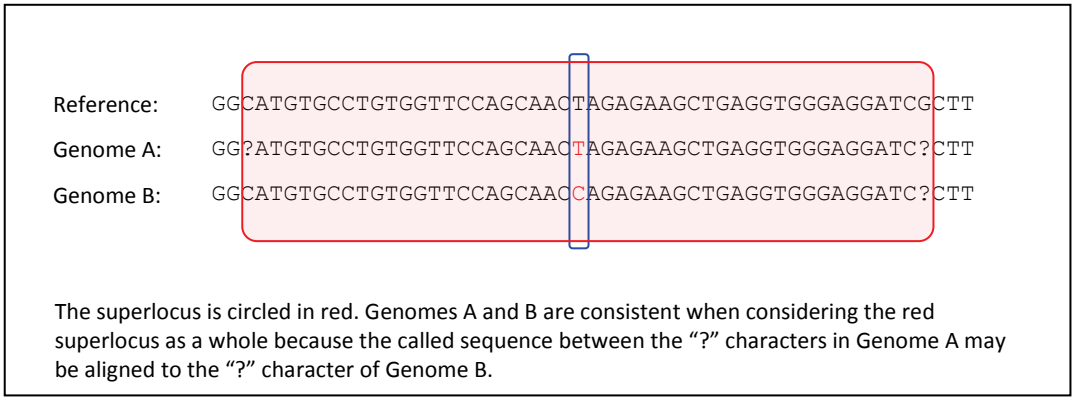
calldiff

The calldiff tool compares two variant files to determine where the two genomes differ, and how. To achieve this, it first gathers variants into superloci, which may account for several nearby variants. It compares the genomes for each superlocus then refines the comparison result to get call-level and locus-level detail. For example, it can be used to help find potential somatic mutations in a tumor-normal comparison, or to find where two assemblies of the same genome differ.

calldiff is less sensitive than snpdiff to the canonical alignment.

If the superloci are too small, superlocus comparison tends to be overly sensitive to canonical alignment. But if superloci are too large, superlocus comparison tends to allow any sequence from one genome to match in a gap of unknown sequence in the other genome. As an example of a superlocus that is too large, suppose we had the sequence from a haploid chromosome of two genomes shown in Figure 8:

Figure 8: Example of a Superlocus that is too Large



The superlocus is circled in red. Genomes A and B are consistent when considering the red superlocus as a whole because the called sequence between the "?" characters in Genome A may be aligned to the "?" character of Genome B.

When considering the red superlocus in Figure 3, and when interpreting the meaning of the calls literally, we can see that all the called bases between the "?" characters in Genome A may be aligned to the "?" character of Genome B, and the genomes are consistent. But when considering the blue box to be the superlocus, we see that the genomes are inconsistent. In different contexts, one superlocus or the other may be preferable, but generally for most comparisons, we would want a comparison algorithm in this case to state the inconsistency between the genomes. To achieve this, a comparison algorithm must either be very precise about how to compare superloci or very precise about how to define a superlocus:

- *Precise about how to compare superloci:* such as when using the red superlocus, determine that there is enough high complexity and uncommon sequence between the "?" characters in Genome A that the SNP in the middle must be aligned as called.
- *Precise about how to define a superlocus:* such as always use the blue superlocus in this situation.

calldiff achieves its specificity by being precise about its superlocus definition.

To determine the superloci, calldiff begins by labeling each reference region containing a variant in either variant file as a superlocus. The superloci are then extended according to the following criteria:

1. **Circular prefix/suffix matching.** For every call whose *alleleSeq* does not contain "N" or "?", do prefix matching to the right along the reference and suffix matching to the left along the reference of both the alleleSeq and the reference sequence, such that the superlocus extension does not exceed P bases (the P limit is necessary to limit the superlocus size for pathological situations). For example, if the call is for an insertion of "ACGT" and the reference sequence directly to the right is "ACGA", three prefix bases of the alleleSeq can be matched to the reference sequence directly to the right, indicating that an equivalent insertion exists at each position in that range. So the superlocus must be extended to account for any variants within three bases to the right of the variant. Additionally, in the example above, if the sequence directly to the right of the call was "ACGTACGA", then the entire insertion of four bases can be prefix matched, and continuing along the reference, the next three bases also match the prefix of the insertion. (This is circular prefix matching.) So the superlocus must be extended to the right by seven bases.
2. **Fixed base count.** Always extend superloci to the right and left by N bases, where N is a command-line configurable parameter. Currently, this parameter defaults to 0.
3. **Fixed count of distinct 3-mers.** Always extend by M distinct reference 3-mers to the right and left, where M is a command-line configurable parameter. In regions of low reference sequence complexity, this results in longer superloci. In regions of high reference sequence complexity, this results in shorter superloci. Currently, this parameter defaults to 4.

After the superloci have been fully extended, overlapping and abutting superloci are combined into a single superlocus.

After superloci have been found, all possible phasings consistent with the *hapLink* values in the calls are used to produce hypotheses about what the genome sequence is, for each variant file. Then each permutation of each hypothesis (one permutation for haploid, two for diploid, and six for triploid) is compared to each hypothesis of the other variant file according to a literal interpretation of their sequence. In other words, any number of bases may align against length no-calls ("?"). The best comparison is produced, such that the number of discordant haplotypes is minimized. The alleles of the best comparison are then segmented to get call-level comparison results. The call-level comparison results are defined to be no worse than the result for the allele as a whole; if a segment comparison results in a worse comparison result than the allele as a whole, the allele's comparison result is used in its place. The call-level comparison results are then used to classify the comparison of each locus as a whole.

The results of calldiff are, for each allele, a comparison classification as described in Table 1.

Table 1: Classification of Comparison Results

Classification	Description
ref-identical	The alleles of the two variant files are identical, and they are consistent with the reference.
alt-identical	The alleles of the two variant files are identical, and they are inconsistent with the reference.
ref-consistent	The alleles of the two variant files are consistent, and they are consistent with the reference.
alt-consistent	The alleles of the two variant files are consistent, and at least one is inconsistent with the reference.
onlyA	The alleles of the two variant files are inconsistent, but only file A is inconsistent with the reference.
onlyB	The alleles of the two variant files are inconsistent, but only file B is inconsistent with the reference.
mismatch	The alleles of the two variant files are inconsistent with each other and with the reference.
phase-mismatch	The two variant files would be consistent if the *hapLink* field had been empty, but the *hapLink* entry causes them to be inconsistent.
ploidy-mismatch	The superlocus did not have uniform ploidy.

For non-haploid superloci, the comparison results for the alleles are joined by a semi-colon. For example, for a diploid hypothesis where variant file A calls reference and variant file B calls a het SNP, you might have a comparison result that looks like “ref-identical;onlyB”.

For example, suppose we use calldiff to compare a tumor genome (file A) and a normal genome (file B) from the same individual. We can find purported somatic mutations by looking for “ref-identical;onlyA”. We can find purported loss of heterozygosity (LOH) by looking for “ref-identical;onlyB” or “alt-identical;onlyA”. We might expect fewer superloci classified as “alt-identical;onlyB”, as the likely reason for this is assembly error – overcall in the normal genome.

calldiff for scoring somatic variations (beta)

When calldiff is used to find differences between two genomes from the same individual, such as a tumor/normal pair, it can be useful to know the likelihood that any given variation discovered in genome A (the tumor) but not in the genome B (the normal) is truly somatic, as opposed to being caused by a false negative error in the genome B or a false positive error in the genome A. calldiff can produce a report that lists all variations that are present only in the first genome, annotated with a score that indicates the somatic likelihood ratio.

The report includes simple loci that are fully called and can be classified as homozygous, heterozygous, or haploid SNP, DEL, INS, or SUB. The report does not include any variations in the mitochondrial DNA nor in the regions where the expected ploidy doesn’t match between two genomes (that is, most of chrX when comparing male and female genomes).

Different variation types such as het-SNP versus hom-SNP versus hom are scored separately. The scores can be used to order the variations of a given type from least likely to be somatic (lower, usually negative scores) to most likely to be somatic (higher, usually positive scores). The score computation is based on:

- *For SNPs*: reference score of the locus in the genome B and the reference score of the locus in the genome A, or
- *For all other types of variation*: the reference score of the locus in the genome B and the evidence score of the variation in genome A.

Because the variation score indicates the likelihood of the variation being exactly identical to the one described in the assembly, we find the reference score (which estimates the likelihood that the sequence is not identical to the reference in the first genome) to be a better measure for somatic variation.

An example of the SomaticOutput is as follows:

Table 2: Example of Somatic Output

SuperlocusId	LocusClassification	locus	ploidy	haplotype	chromosome	begin	end	varType	reference	alleleSeq	totalScore	hapLink	xRef	VarScoreA	RefScoreB	SomaticScore
443	hom-snp	163	2	1	chr1	888	889	snp	G	T	113			-306	63	-8
679	het-ins	180	2	1	chr1	1018	1018	ins		A	51			51	100	-23
720	hom-snp	186	2	1	chr1	1078	1079	snp	C	A	158			-1803	273	1
984	hom-del	211	2	1	chr1	1292	1294	del	AG		94			2846	107	6
986	hom-sub	212	2	1	chr1	1292	1295	sub	AAT	TGA	99			2846	-250	-43

The report contains calls from the variant file of file A, annotated with the superlocus ID, locus classification, the score used in the somatic score analysis for file A (as described above – the reference score for SNPs, the evidence score for other variant types), the reference score for file B, and the somatic score.

To produce this data, calldiff requires access to the roots of the intact export packages for both genomes.

Computing Somatic Scores

This explanation of how calldiff computes somatic scores assumes that the first genome is a tumor and the second one is the corresponding normal sample.

Definitions:

- Q — ratio of the probability of the discordant calls indicating a true somatic event to the probability of the discordance being caused by a wrong call in either of the assemblies.
- R_N, R_T, V_N, V_T — possible states of the genomes at a given locus; respectively, normal genome is equal to reference, tumor genome is equal to reference, normal genome has the variation, tumor genome has the variation.
- r_N — reference score of the locus in the normal genome.
- s_T — score that is most informative of the presence of the variation in the tumor genome (reference score of the locus for SNPs or evidence score for other variations).
- D — prior condition of the discordance between the calls in the normal and tumor genomes.

Given these definitions, the reciprocal of the desired probability ratio can be written as:

$$\frac{1}{Q} = \frac{P(R_N R_T | r_N, s_T, D) + P(V_N V_T | r_N, s_T, D)}{P(R_N V_T | r_N, s_T, D)}$$

Assuming that the states of reference and tumor genome at a given locus are independent, we get:

$$\frac{1}{Q} = \frac{P(R_N | r_N, s_T, D) P(R_T | r_N, s_T, D) + P(V_N | r_N, s_T, D) P(V_T | r_N, s_T, D)}{P(R_N | r_N, s_T, D) P(V_T | r_N, s_T, D)}$$

Because the state of the normal genome is unlikely to depend on the score of the tumor (and vice versa), this can be simplified to:

$$\frac{1}{Q} = \frac{P(R_T | s_T, D)}{P(V_T | s_T, D)} + \frac{P(V_N | r_N, D)}{P(R_N | r_N, D)}$$

And, by applying Bayes' rule, we get:

$$\frac{1}{Q} = \frac{P(s_T | R_T, D) P(R_T | D)}{P(s_T | V_T, D) P(V_T | D)} + \frac{P(r_N | V_N, D) P(V_N | D)}{P(r_N | R_N, D) P(R_N | D)}$$

To estimate this quantity, we use the following simple lemmas:

- $P(s_T | V_T, D) \approx P(s_T | V_T)$ — We assume that the tumor scores in the discordant loci are distributed approximately the same as the scores of all true variations in the tumor.
- $P(r_N | R_N, D) \approx P(r_N | R_N)$ — In a similar manner, the reference scores over discordant loci are assumed to be drawn from the overall distribution of the reference scores in the true reference regions of the normal genome.
- $P(s_T | D) \approx P(s_T | R_T, D)\, P(R_T | D) + P(s_T | V_T, D)\, P(V_T | D)$ — At the given locus, the true state of the genome is assumed to be either reference or the called variation. Hence, $P(s_T | R_T, D)\, P(R_T | D) \approx P(s_T | D) - P(s_T | V_T, D)\, P(V_T | D)$
- In a similar manner, $P(r_N | V_N, D) P(V_N | D) \approx P(r_N | D) - P(r_N | R_N, D) P(R_N | D)$

We can rewrite the overall expression as:

$$\frac{P(s_T|D)}{P(s_T|V_T)P(V_T|D)} + \frac{P(r_N|D)}{P(r_N|R_N)P(R_N|D)} - 2$$

Now we estimate the probabilities $P(V_T|D)$ and $P(R_N|D)$. We again assume that $P(s_T|D) \approx P(s_T|R_T, D)\, P(R_T|D) + P(s_T|V_T, D)\, P(V_T|D)$, and consequently:

$$P(V_T|D) \approx \frac{P(s_T|D)}{P(s_T|V_T, D)} - \frac{P(s_T|R_T, D)\, P(R_T|D)}{P(s_T|V_T, D)}$$

This holds for all score values s_T. However, when the variation score is high, the ratio $\frac{P(s_T|R_T,D)}{P(s_T|V_T,D)}$ is very low (that is, a variation with a high score is much more likely to be correct than to be a false positive), and the contribution of the second term can be ignored. Because the overall expression is constant in s_T, the first term must reach its minimum for the same value of the score as the second term reaches its minimum value of (approximately) zero, so we can conclude that:

$$P(V_T|D) \approx \min_{s_T} \frac{P(s_T|D)}{P(s_T|V_T, D)} \approx \min_{s_T} \frac{P(s_T|D)}{P(s_T|V_T)}$$

The second simplification depends on the same assumption about the distribution of variation scores that we already used previously.

Similar logic can be applied for $P(R_N|D)$. The overall expression becomes:

$$\frac{1}{Q} \approx \frac{\frac{P(s_T|D)}{P(s_T|V_T)}}{\min_{s_T} \frac{P(s_T|D)}{P(s_T|V_T)}} + \frac{\frac{P(r_N|D)}{P(r_N|R_N)}}{\min_{r_N} \frac{P(r_N|D)}{P(r_N|R_N)}} - 2$$

The actual score as written to the report file is:

$$score = 10 \log_{10} Q = -10 \log_{10} \frac{1}{Q}$$

The distributions $P(s_T|D)$ and $P(r_N|D)$ are estimated separately for each of the eight (het/hom, snp/del/ins/sub) locus types by binning the scores in such a way that each bin contains at least 50 loci. The separation is required because the reference score and evidence score distributions differ significantly between these variation types. If the total count of events is below 50, calldiff creates only one bin and produces degenerate scores for that variation type (that is, all variations of that type will get the same score).

The distribution $P(s_T|V_T)$ is estimated for a given variation type by using the loci where both assemblies make the same variation call of that type. We use the same bin score boundaries as we used for the corresponding $P(s_T|D)$. calldiff estimates the distribution $P(r_N|R_N)$ using the loci where both assemblies agree on calling the reference for at least 50 bases in either direction.

The minima required by the score formula are computed by taking the minimum ratio across all score buckets. The somatic score is therefore less informative when the total number of score buckets is small, and it conveys no information at all when there is only one bucket for a locus type.

Somatic Score and Quality

The performance of the somatic score was evaluated by comparing an ATCC adenocarcinoma cell line (NCI-H1395) to a matching normal cell line (NCI-BL1395, lymphoblast). The null distribution of scores in the absence of any true somatic variations was estimated by running the tool on a pair of replicate libraries built from a HapMap cell line (NA19240). The results, plotted in Figure 9, show that the score can be used to control the trade-off between sensitivity and specificity in somatic variation detection.

Figure 9: Performance of calldiff Somatic Scoring

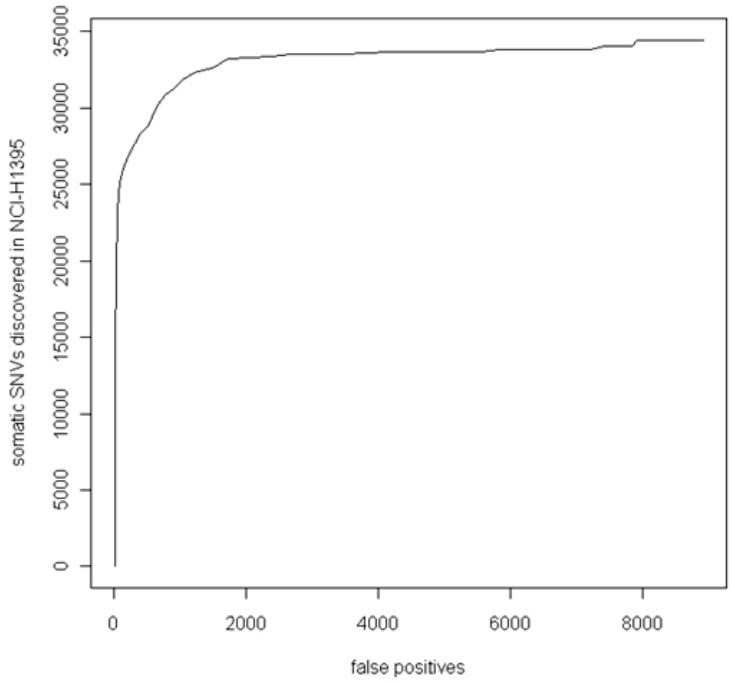


Table 2 lists the number of apparent somatic variations observed at various thresholds.

Table 3: False Positive and Total Discovered Somatic SNVs at Various Somatic Score Thresholds

Somatic Score Threshold	False Positives (Genome-wide, estimated from NA19240 replicates)	Somatic SNVs discovered in NCI-H1395
0	6	7,857
-5	37	21,994
-10	142	26,191
-15	510	28,846
-20	1217	32,343
-25	2665	33,498
-30	5267	33,708
-50	8129	34,460
-100	8916	34,460

The data in Table 2 suggest that the false discovery rate (FDR) of somatic SNVs may be controlled using the somatic score. In this example, assuming similar error characteristics in the two pairs of genomes would lead to an FDR estimate of < 1% at a somatic score threshold of -10 dB. The trade-off between sensitivity and specificity achievable in any particular tumor-normal comparison would vary depending on several factors: the extent of aneuploidy, impurity, and heterogeneity within the tumor sample, the coverage characteristics of the two genomes, and the number of true somatic events.

Many-Genome Comparison: listvariants (beta)

The superlocus approach to genome comparison achieves a good combination of sensitivity to genomic variation and insensitivity to canonical alignment for a small number of genomes. But as the number and variety of genomes grows, superloci can grow arbitrarily large. As shown in "calldiff," when superloci grow too large, the literal interpretation of sequence compatibility employed by calldiff tends to be insensitive to real genomic differences.

cgatools supports many-genome comparison through the combination of two tools: listvariants and testvariants. The listvariants command lists all the genomic mutations found in an arbitrary number of genomes, and the testvariants command tests each of those mutations against a set of genomes to determine the compatibility of the genomes to each mutation.

The listvariants command merges and lists all the fully called mutations from a set of variant files (that is, each line from each variant file that is fully called and inconsistent with the reference). listvariants is as specific as it can be about mutations without splitting up called mutations from the variant file. For example:

Reference:	CGAATTACAT
Allele 1:	CGCATTATAT
Allele 2:	CGAATTACAT

In this case, suppose the variant file listed this sequence as two SNP mutations with the same *hapLink* to indicate they are on the same haplotype. In this case, listvariants also lists the two SNPs separately. In this way, the many-genome comparison can be very specific about where genomes differ, although it loses information about which variants occurred on the same haplotype as other variants.

listvariants also canonicalizes any input variants it encounters before writing them to the output. It uses the rightmost variant that is equivalent to the input variant. For example:

Reference:	CG-AAAAA-CAT
Alternative 1:	CG-AAAAAACAT
Alternative 2:	CGAAAAAA-CAT

The two alternatives above have the same sequence, but have different alignments against the genome. If the input genomes list both insertions, the alternative 2 alignment is canonicalized (transformed) into the alternative 1 alignment because it is the rightmost alignment that describes an equivalent sequence. The two variants are then merged as equivalent, and a single output record is produced.

The output records retain the annotations present in the input variant file. Figure 9 shows an example:

Figure 10: Output records retain input annotations

variantId	chromosome	begin	end	varType	reference	alleleSeq	xRef
1034	chr1	972803	972804	snp	T	C	dbsnp:rs3128102
1035	chr1	972856	972857	snp	T	C	dbsnp:rs10267
1036	chr1	975024	975025	snp	G	T	
1037	chr1	975128	975129	snp	C	T	dbsnp:rs2275813
1038	chr1	975311	975313	sub	GG	A	dbsnp:rs56255212
1039	chr1	975322	975323	snp	T	C	dbsnp:rs2275811
1040	chr1	975371	975372	snp	G	A	
1041	chr1	975900	975901	snp	G	A	

Many-Genome Comparison: testvariants (beta)

The testvariants command processes the variants listed by listvariants and writes each input record to the output, along with a flag for each allele of each genome to indicate if the variant is present on that allele. Table 3 lists set of possible flags for each allele:

Table 4: testvariants Flags

Flag	Description
0	The allele is inconsistent with the variant.
1	The allele is fully called, and is consistent with the variant.
N	The allele has no-calls, and the allele is consistent with the variant.

For example, for the listvariants example in Figure 11, the testvariants output against three genomes is shown in Figure 12:

Figure 11: testvariants Output against Three Genomes

```
variantId  chromosome  begin   end     varType  reference  alleleSeq  xRef             ASM1  ASM2  ASM3
1034       chr1        972803  972804  snp      T          C          dbsnp:rs3128102  11    01    11
1035       chr1        972856  972857  snp      T          C          dbsnp:rs10267    11    11    11
1036       chr1        975024  975025  snp      G          T                           00    01    NN
1037       chr1        975128  975129  snp      C          T          dbsnp:rs2275813  00    01    00
1038       chr1        975311  975313  sub      GG         A          dbsnp:rs56255212 11    1N    1N
1039       chr1        975322  975323  snp      T          C          dbsnp:rs2275811  11    01    1N
1040       chr1        975371  975372  snp      G          A                           00    01    00
1041       chr1        975900  975901  snp      G          A                           1N    00    01
```

In this example, ASM1 and ASM3 are homozygous for variant 1034, but ASM2 is heterozygous. ASM1 does not have variant 1036, but ASM2 is heterozygous for the variant, and ASM3 is no-called at that position.

The testvariants command tests each variant against each genome independently. To do so, it first constructs a one-genome superlocus to keep track of which loci may be used in the comparison. The superlocus is constructed using the same superlocus rules as calldiff, except the "Fixed count of distinct 3-mers" used is 6 instead of 4.

After a superlocus has been found, all possible phasings consistent with the *hapLink* values in the calls are used to produce hypotheses about what the genome sequence is. testvariants chooses the phasing that results in the most 1's, then the most 0's.

To test a particular phasing, for each allele, testvariants first finds a base set of calls consisting of the minimal set of calls overlapping the variant, extended to the left and right according to the prefix/suffix matching rule of calldiff. Then testvariants compares the call sequence to the variant sequence (extended to the left and right by reference sequence), for every sequential sequence of calls in the superlocus that covers the base set of calls. This results in "1" if any compared sequence matches the variant sequence, or "N" if any compared sequence is compatible with the variant sequence but contains no-calls. Otherwise, the result is "0".

Although the testvariants algorithm may achieve a reasonable middle ground between sensitivity to real genomic variation and insensitivity to canonical alignment for the many-genome comparison problem, the following limitations apply:

- testvariants is more sensitive to the canonical alignment than calldiff. As such, it is not the ideal tool for comparing a small number of genomes (such as 2 or 3).
- For simplicity of output file format and interpretation, testvariants does not transfer the score of the variant calls or the reference scores in the input genomes to the output file. Not having scores further limits the testvariants output for use in analyzing the genomic differences of a small number of genomes.

Format Conversion Tools

The primary goal of the Complete Genomics export formats is to represent the data in a concise and simple way. As such, they are not always the best formats for doing certain kinds of data processing. Moreover, some users have existing programs that expect inputs in various other data formats. As a result, **cgatools** aims to provide data conversion capabilities.

map2sam

The map2sam tool converts Complete Genomics exported reads and initial reference mappings to the SAM format. For pipelines that require eventually converting to the BAM format, the output of map2sam can be sent to standard output, which can be processed by SAM Tools. For example, this command pipeline creates an indexed, reference-sorted BAM file:

```
cgatools map2sam --reads=/path/to/reads.tsv.bz2 \
                 --mappings=/path/to/mappings.tsv.bz2 \
                 --library=/path/to/lib_DNB.tsv | \
                 --reference=/path/to/build36.crr | \
    samtools view -uS - | \
    samtools sort - result && samtools index result.bam
```

Complete Genomics reads are initially mapped to the reference genome using a fast algorithm, and these initial mappings are later both expanded and refined by a form of local *de novo* assembly applied to putatively variant regions of the genome.

IMPORTANT: The map2sam tool converts the initial reference mappings, and not the additional mappings to variants discovered during the assembly process.

The following additional limitations apply to map2sam output:

- The converted mappings are reference mappings only. The mappings used as evidence to make indel calls are not included.
- SAM does not have strong support for overlapping sub-reads (for example, the negatively sized intra-read gaps), which are present in Complete Genomics data. To represent overlapping reads, the strongest base call is put in the SAM mapping record, and the alternative base calls are represented using the GS/GQ/GC tags of the mapping record.
- The SAM validator provided by the Picard project (picard.sourceforge.net) does not allow specifying a primary mapping for reads that do not have consistent mates. As a result, only reads which have consistent mate pair mappings have a mapping marked as the primary mapping record (mapping record with the "not primary" FLAG, 0x0100, set to 0).
- The NM tag (edit distance to reference sequence) is not currently produced by map2sam.
- The R2 and Q2 tags (mate sequence and quality scores) can be generated optionally.

Representation of the Complete Genomics data in the SAM output

The detailed descriptions of the map2sam output below assume some familiarity with Complete Genomics data and terminology. We recommend you consult the Complete Genomics Data File Formats document and FAQs if you are mostly familiar with other Next-Gen platforms. These documents can be obtained from support@completegenomics.com. Additionally, the Complete Genomics assembly process and some of its implications are described in the Complete Genomics technology whitepaper and in more detail the *Science* paper. We recommend you consult the Complete Genomics FAQ documents in considering how to best use these data. See "References" for more information on these documents.

This description is based on the SAM Format Specification described in "References."

Table 5: Header Fields

Section	Tag	Value	Description	Example
@HD	VN	0.1.2	Version of SAM spec	
	SO	"DnbId sorted"	Sort Order. **Note:** A DNB is a clone.	
@SQ	SN	*ChromosomeName*	Sequence Name. Included for all chromosomes in the reference genome.	SN:chr1
	LN	*ChromosomeLength*	Length of the chromosome. Included for all chromosomes in the reference genome.	LN:247249719
	UR	*ReferenceFilePath*	Path to the input reference file	UR: reference.crr
	AS	*ASM ID*	CGI Data Analysis Pipeline Run ID	AS:GS19240-ASM
@RG	ID	*LaneID*	Slide and Lane ID	ID:GS08081-FS3-L02
	SM	*SampleID*	Sample Id	SM:GS00028-DNA-C01
	LB	*LibraryID*	Library ID of the library	LB:GS00433-CLS
	PU	*LaneID*	Slide and Lane ID	PU:GS08081-FS3-L02
	CN	"Complete Genomics"	Name of sequencing center producing the read	
	DT	*ExportDate*	The CGI data analysis timestamp stored in the CGI reads file.	DT:2010-01-21
	PL	"Complete Genomics"	Platform/technology used to produce the read.	
@PG	ID	"cgatools"	Program name	
	VN	*Version*	Program version	VN:0.5.0
	CL	*Command line*	Command line	Complete string

Table 6: Mapping Record Fields

Field	Value	SAM Definition	map2sam Usage
QNAME	*SlideID-LaneID*: *DnbOffest*	Query Name	The QNAME value is constructed from the full lane Id (SlideId+LaneId) and 0-based DNB offset from the beginning of the Reads file provided as input. For example: GS08081-FS3-L02-3:244
FLAG	0x0001	The read is paired in sequencing.	The flag is always set for CGI data. The current CGI technology always produces paired reads.
	0x0002	The read is mapped in a proper pair.	There are two possible behaviors: 1) The flag is set only if the other HalfDNB is mapped consistently. That is, within the valid range of the mate gap and on the same strand as the current HalfDnb. 2) The case is similar to the previous one but also includes the situation when the other HalfDNB is mapped in a random place but both HalfDNBs have unique mappings within the whole genome. In that case the mate HalfDNB will be treated as a mate even though it is not consistent.
	0x0004	The query sequence itself is unmapped.	The flag is set when there are absolutely no mappings found for this HalfDNB.
	0x0008	The mate is unmapped.	The flag is set only when there are no mappings found for this HalfDNB's mate.
	0x0010	Strand of the query	0 for forward; 1 for reverse strand.
	0x0020	Strand of the mate	0 for forward; 1 for reverse strand.
	0x0040	The read is the first read in a pair.	The flag is set if the current HalfDNB is from the 5' end of the original cloned insert.
	0x0080	The read is the second read in a pair.	The flag is set if the current HalfDNB is from the 3' end of the original cloned insert.
	0x0100	The alignment is not primary.	The flag is set if there is a better mapping of the same HalfDNB having higher value of *MAPQ* (see "MAPQ" in this table) or the other HalfDNB is not mapped.
	0x0200	The read fails platform/vendor quality checks.	Always set to 0.
	0x0400	The read is either a PCR duplicate or an optical duplicate.	Always set to 0.
RNAME	*ChromosomeID* or "*"	Reference sequence NAME	Can be "*" if this HalfDNB doesn't have mappings.
POS	*Current Mapping Position* or 0	1-based leftmost POSition/coordinate of the clipped sequence	The position reported in a Mappings file record from a CGI export package offset by 1 (CGI export format reports mapping positions 0-based). 0 is reported if there are no mappings found for this HalfDNB.

Field	Value	SAM Definition	map2sam Usage
MAPQ	*CG_Mapping weight*	MAPping Quality (Phred-scaled probability that the mapping position of this read is incorrect.)	The probabilities are reported in different ranges for consistent pair reads (Flag 0x0002, case 1) and for non-paired mappings. CGI does not recommended that values of consistent and inconsistent mappings be directly compared.
CIGAR	C*igarString* and GS/GQ/GC flags	Extended CIGAR string	Currently, CGI initial mappings files do not allow insertions or deletions. Therefore, only M and N operations are used in the *CIGAR* field. The negative gaps are represented using GS/GQ/GC flags. The CIGAR sequence will represent the positive gaps using N and ignore the negative gaps.
MRNM	"=" or *ChromosomeID* or "*"	Mate reference sequence NaMe; "=" if the same as *RNAME*	Reports "*" if there is no consistent mate found.
MPOS	*MatePosition* or 0	1-based leftmost mate POSition of the clipped sequence	Reports 0 if there is no consistent mate found.
ISIZE	*DistanceToMate* or 0	Inferred Insert SIZE	The distance between the consistent mate start position and the start position of the current HalfDNB mapping. The value is 0 if the mates are mapped to different chromosomes.
SEQ	*Sequence*	Query SEQuence; "=" for a match to the reference; n/N/. for ambiguity	The regions of overlapping bases in the negative gaps contain the bases with higher scores.
QUAL	*QualityScores*	Query QUALity; ASCII-33 gives the Phred base quality	The values are copied from the corresponding record of the CGI Reads file.
TAG	GS/GQ/GC R2/Q2	Tags	GS/GQ/GC flags are used to represent CGI-specific negative wobble gaps in HalfDNBs. See SAM Format Specification in "References" for the description of the flags. The optional tags that are provided are R2,Q2.

Rules to Set the "not primary" Flag (0x0100)

The flag "not primary" is set for a HalfDNB mapping in the following cases:

- There is another mapping of the same HalfDNB having a higher *MAPQ* value.
- The mapping of a HalfDNB doesn't have a consistent mate pair mapping, and there are mappings found for the mate HalfDNB.
- The mapping's best mate has a best mate that is not the current mapping.

Combining Mapping Records in SAM

1. The best mapping pair (the best score) of a DNB is reported with the "non-primary" flag set to 0. Both mappings should refer to each other as the best mates.
2. All the other mappings of that DNB are reported in non defined order and have the "non-primary" flag set to 1.

3. If both HalfDNBs are mapped uniquely but not consistently, they are not reported as primary ("non-primary" flag is set to 0) even though they are not consistent mates.
4. If only one HalfDNB is mapped, the best mapping of that HalfDNB is reported followed by a "non-mapped" mapping record of the other HalfDNB. The alignment position of the other HalfDNB is set to the same values as the mapped HalfDNB and the "non-primary" flag of the records is set to 0. The not mapped record will be marked as "not mapped" by appropriate flags.
5. All the other mappings of the mapped read from number 4 above are reported one record per mapping having "non-primary" flag set to 1.
6. All the not mapped reads are reported in pairs aligned to the 0 position. The alignment position is important to keep the records together while sorting and merging BAM files.

evidence2sam (beta)

The evidence2sam tool converts Complete Genomics evidence mappings to the SAM format. The current implementation is in beta form. For pipelines that require eventually converting to the BAM format, the output of evidence2sam can be sent to standard output, which can be processed by SAM Tools. For example, this command pipeline creates an indexed, reference-sorted BAM file:

```
cgatools evidence2sam \
        --beta \
        --evidence-dnbs=/path/to/evidenceDnbs-chrN-XXX.tsv.bz2 \
        --reference=/path/to/build36.crr | \
    samtools view -uS - | \
    samtools sort - result && samtools index result.bam
```

Complete Genomics evidence mappings are the mappings that were used to call variations found by the CGI genome assembly process. The assembly process uses a local *de novo* method to find likely alleles for a variation interval (small region of the genome, typically less than 200 bases), then an optimization process to refine the allele choices. The evidence mappings are DNB alignments that indicate support for the best hypothesis found during the assembly process. The evidence2sam tool can be used to convert these mappings to SAM for visualization in a genome browser like IGV. The details of the Complete Genomics data representation in the SAM output are covered in the map2sam tool description in this document.

In two situations, a DNB may have multiple mapping records present in the evidence DNB mappings provided by Complete Genomics. First, if the best hypothesis is heterozygous and contains two non-reference alleles, support is also given for the reference allele. In this case, if a DNB supports two of the three alleles equally well (or similarly well) and not the third allele, then the evidence DNB mappings contain a record showing alignment of the DNB to each of the two alleles it supports. Second, if there are two regions of the genome with similar sequence such that DNBs align well to either sequence, those DNBs may be used as evidence for alleles in both regions. A post-processing step of the CGI assembly process finds such regions and no-calls them.

Because most tools that visualize SAM do not have rich features to specify an allele a DNB maps against, visualization of the duplicate mapping records present in the evidence can be confusing. For this reason, the evidence2sam tool has an option to de-duplicate the mappings present in the evidence, for both forms of duplication described above, for duplicate DNB mappings that are nearby on the reference. Specifically, the evidence2sam tool de-duplicates using the following algorithm, for each variation interval:

1. Update the read-ahead buffer to ensure it contains all evidence mapping records up to 1 Kb to the right of the position of the rightmost evidence mapping record for this interval.
2. Moving from position 0 to the end of the chromosome, processing each mapping record of the current variation interval as follows:
    a. Collect all the mappings of the same DNB that belong to the current interval or mappings from different intervals that overlap the corresponding arm/both arms of the selected DNB.

b. Run one-DNB de-duplication. This operation deletes all the collected DNB mappings from the buffer except the "best" one.

c. Write the "best" mapping into the SAM output stream.

d. Remove the "best" mapping from the buffer and proceed to the next mapping in the current interval.

e. If the last mapping in the current interval has been processed, remove the mappings processed for the current interval from the read-ahead buffer.

During de-duplication, the following rules are used to determine the best mapping for a DNB:

1. If several mapping records belong to the same variation interval, leave only the record that has maximum mapping quality.
2. If several mapping records belong to adjacent variation intervals (same side and strand), leave only the record that has maximum mapping quality.
3. If there are only two mapping records in the set and their different arms support different intervals, construct a composite mapping inheriting MAPQ, position in the reference, and reference alignment from a corresponding mapping record.
4. If there is still more then one mapping record in the input set, select the mapping with highest MAPQ and remove the other mappings.

The following additional limitations apply to evidence2sam output:

- The converted evidence support mappings are the mappings that belong only to the regions where variations were called.
- SAM does not have strong support for overlapping sub-reads (e.g., the negatively sized intra-read gaps), which are present in Complete Genomics data. To represent overlapping reads, the strongest base call is put in the SAM mapping record, and the alternative base calls are represented using the GS/GQ/GC tags of the mapping record.
- When the option to de-duplicate mapping records is not used, evidence2sam reports all mappings as non-primary mappings.
- The NM tag (edit distance to reference sequence) is not currently produced by map2sam.
- The R2 and Q2 tags (mate sequence and quality scores) can be generated optionally.

Delimited File Manipulation Tools

Most files used as input or output for **cgatools** are simple tab-delimited files that can be interpreted as tables. As such, **cgatools** provides tools that manipulate the files as tables.

join (beta)

The join tool works like a database join to merge the results of two delimited input files. It can be used, for example, to annotate the variant file with your own set of annotations. For example, suppose you have the following file:

```
chromosome      begin         end         region
chr1            13            23          InterestingRegion1
chr2            19            20          InterestingRegion2
```

You can annotate the variant file from Figure 3 as follows:

```
cgatools join --match=chromosome:chromosome \
    --overlap=begin,end:begin,end \
    --select='a.*,b.region' \
/path/to/var-tsv.bz2 /path/to/annotations.tsv
```

The result is all the records of the variant file that overlapped with your regions of interest, as shown in Figure 13:

Figure 12: join Example Results

>locus	ploidy	allele	chromosome	begin	end	varType	reference	alleleSeq	totalScore	hapLink	xRef	region
5	2	1	chr1	13	13	ins		A	36			InterestingRegion1
5	2	2	chr1	13	13	ins		A	42			InterestingRegion1
6	2	all	chr1	13	22	ref	=	=				InterestingRegion1
7	2	1	chr1	22	24	del	AT		47	1		InterestingRegion1
7	2	2	chr1	22	24	ref	AT	AT	55	2		InterestingRegion1
16	1	1	chr2	18	20	sub	TT	CG	102			InterestingRegion2

To accomplish this, the join tool first reads all of the annotations file (file B) into memory. Then it streams the variant file (file A); for each record of file A, it finds the records of file B that match the user-selected columns or that overlap the record. As a consequence of this implementation, file B must fit into memory, but file A may be arbitrarily large. Additionally, the output records are in the same order as they are found in file A.

To recap, the limitation of the join tool is:

- File B must fit into memory.