ACT GENOMICS ™

Arriba's workflow

Bioinformatics Development

Sandy

2025.01.15-17

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.

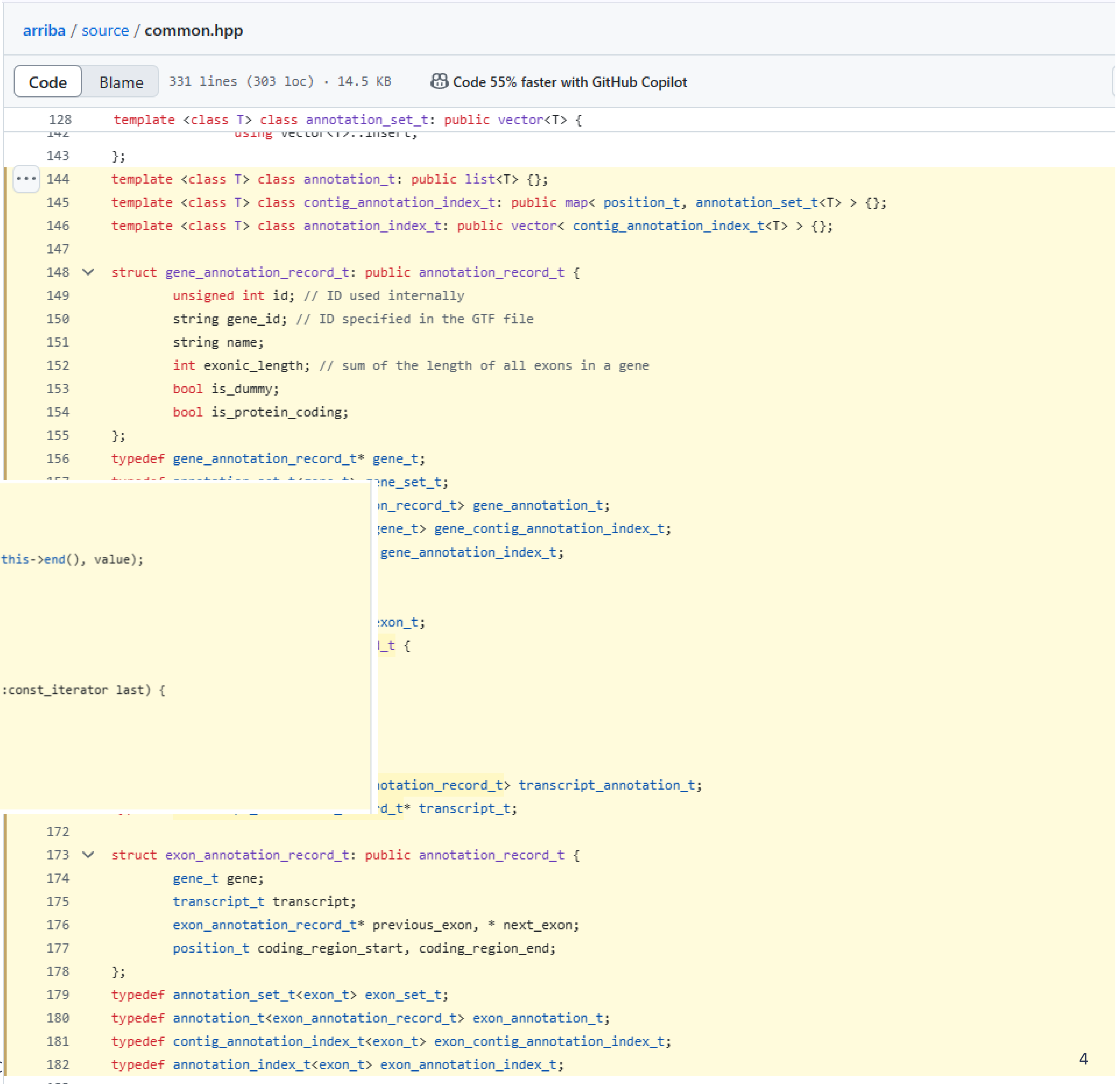
Overview

- How transcript ids are annotated in the Arriba workflow?
- How alternative splicing variants may be annotated?
  (=> Report 1 or 2 RefSeq ID(s) => No specialized algorithm for splicing variants detection.)
- How exon-skipping variants & AR-V7 may be annotated?
  (=> Read-through events => false positive => not included in the fusions.discarded.tsv)
- rMATS (other tool) for splicing variants detection => Not successful

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.

Arriba's annotation

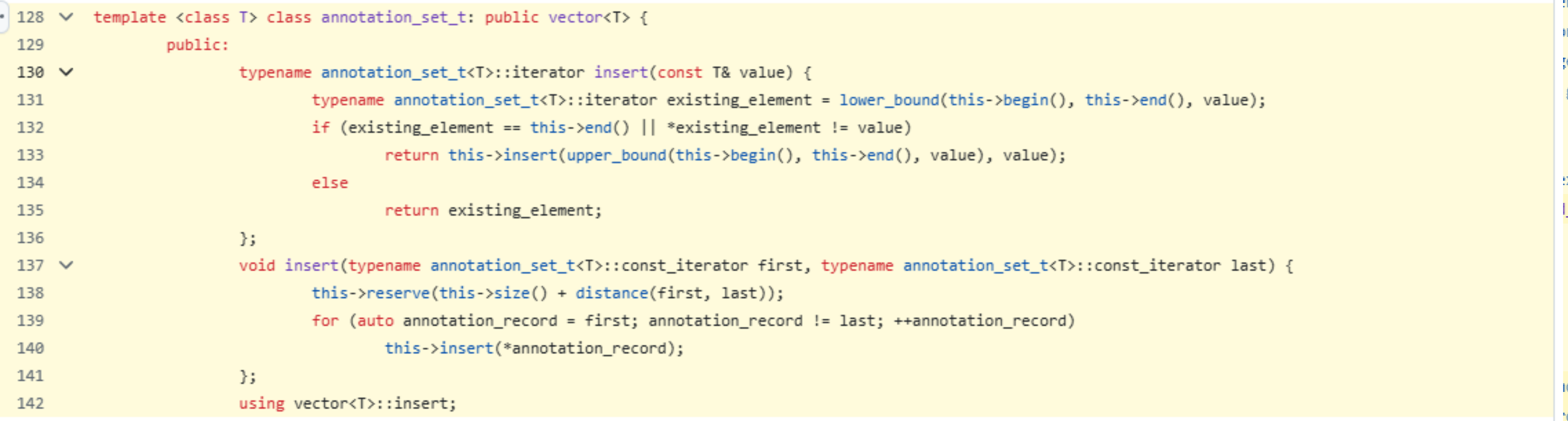
- gtf file parsing (common.hpp)
  => How Arriba extract information from the gtf file



arriba / source / common.hpp

Code Blame 331 lines (303 loc) · 14.5 KB Code 55% faster with GitHub Copilot

```
128 template <class T> class annotation_set_t: public vector<T> {
142                 using vector<T>::insert;
143 };
144 template <class T> class annotation_t: public list<T> {};
145 template <class T> class contig_annotation_index_t: public map< position_t, annotation_set_t<T> > {};
146 template <class T> class annotation_index_t: public vector< contig_annotation_index_t<T> > {};
147
148 struct gene_annotation_record_t: public annotation_record_t {
149         unsigned int id; // ID used internally
150         string gene_id; // ID specified in the GTF file
151         string name;
152         int exonic_length; // sum of the length of all exons in a gene
153         bool is_dummy;
154         bool is_protein_coding;
155 };
156 typedef gene_annotation_record_t* gene_t;
                                           ne_set_t;
                                           n_record_t> gene_annotation_t;
                                           ene_t> gene_contig_annotation_index_t;
                                            gene_annotation_index_t;



                                           xon_t;
                                           _t {





                                           otation_record_t> transcript_annotation_t;
                                           d_t* transcript_t;
172
173 struct exon_annotation_record_t: public annotation_record_t {
174         gene_t gene;
175         transcript_t transcript;
176         exon_annotation_record_t* previous_exon, * next_exon;
177         position_t coding_region_start, coding_region_end;
178 };
179 typedef annotation_set_t<exon_t> exon_set_t;
180 typedef annotation_t<exon_annotation_record_t> exon_annotation_t;
181 typedef contig_annotation_index_t<exon_t> exon_contig_annotation_index_t;
182 typedef annotation_index_t<exon_t> exon_annotation_index_t;
```



```
128 template <class T> class annotation_set_t: public vector<T> {
129         public:
130                 typename annotation_set_t<T>::iterator insert(const T& value) {
131                         typename annotation_set_t<T>::iterator existing_element = lower_bound(this->begin(), this->end(), value);
132                         if (existing_element == this->end() || *existing_element != value)
133                                 return this->insert(upper_bound(this->begin(), this->end(), value), value);
134                         else
135                                 return existing_element;
136                 };
137                 void insert(typename annotation_set_t<T>::const_iterator first, typename annotation_set_t<T>::const_iterator last) {
138                         this->reserve(this->size() + distance(first, last));
139                         for (auto annotation_record = first; annotation_record != last; ++annotation_record)
140                                 this->insert(*annotation_record);
141                 };
142                 using vector<T>::insert;
```

Ref:
https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/common.hpp#L144-L182
https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/common.hpp#L128-L142

ACT GENOMICS ™

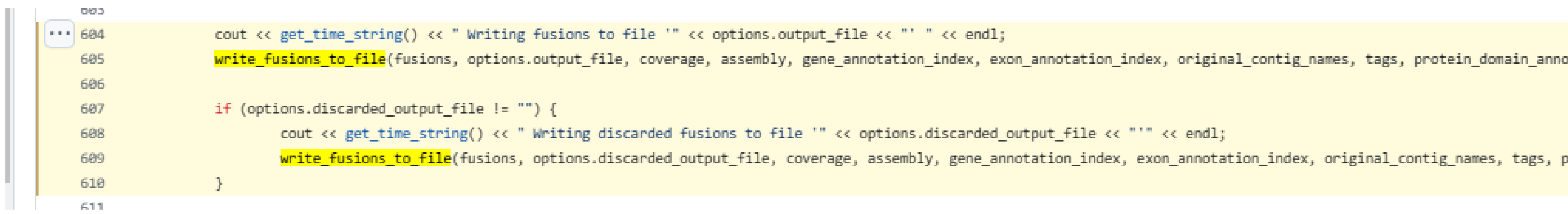
© 2025 ACT Genomics CC

Ref:

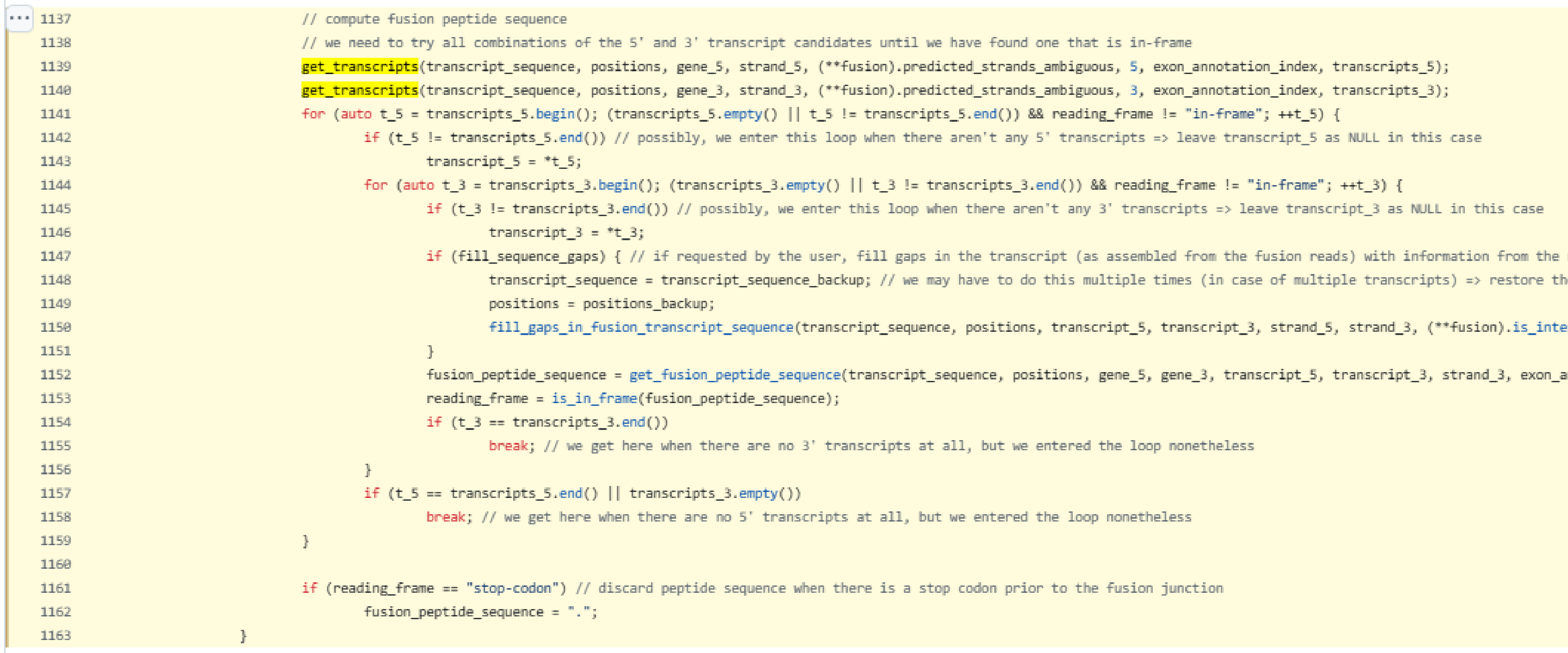
- write_fusions_to_file
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/arriba.cpp#L604-L610
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/output_fusions.cpp#L1137-L1163

Arriba's translation

- write_fusions_to_file (arriba.cpp)
  → main (arriba.cpp)
  → write_fusions_to_file (output_fusions.cpp)
  → get_transcripts (output_fusions.cpp)



```
cout << get_time_string() << " Writing fusions to file '" << options.output_file << "' " << endl;
write_fusions_to_file(fusions, options.output_file, coverage, assembly, gene_annotation_index, exon_annotation_index, original_contig_names, tags, protein_domain_anno

if (options.discarded_output_file != "") {
    cout << get_time_string() << " Writing discarded fusions to file '" << options.discarded_output_file << "'" << endl;
    write_fusions_to_file(fusions, options.discarded_output_file, coverage, assembly, gene_annotation_index, exon_annotation_index, original_contig_names, tags, p
}
```



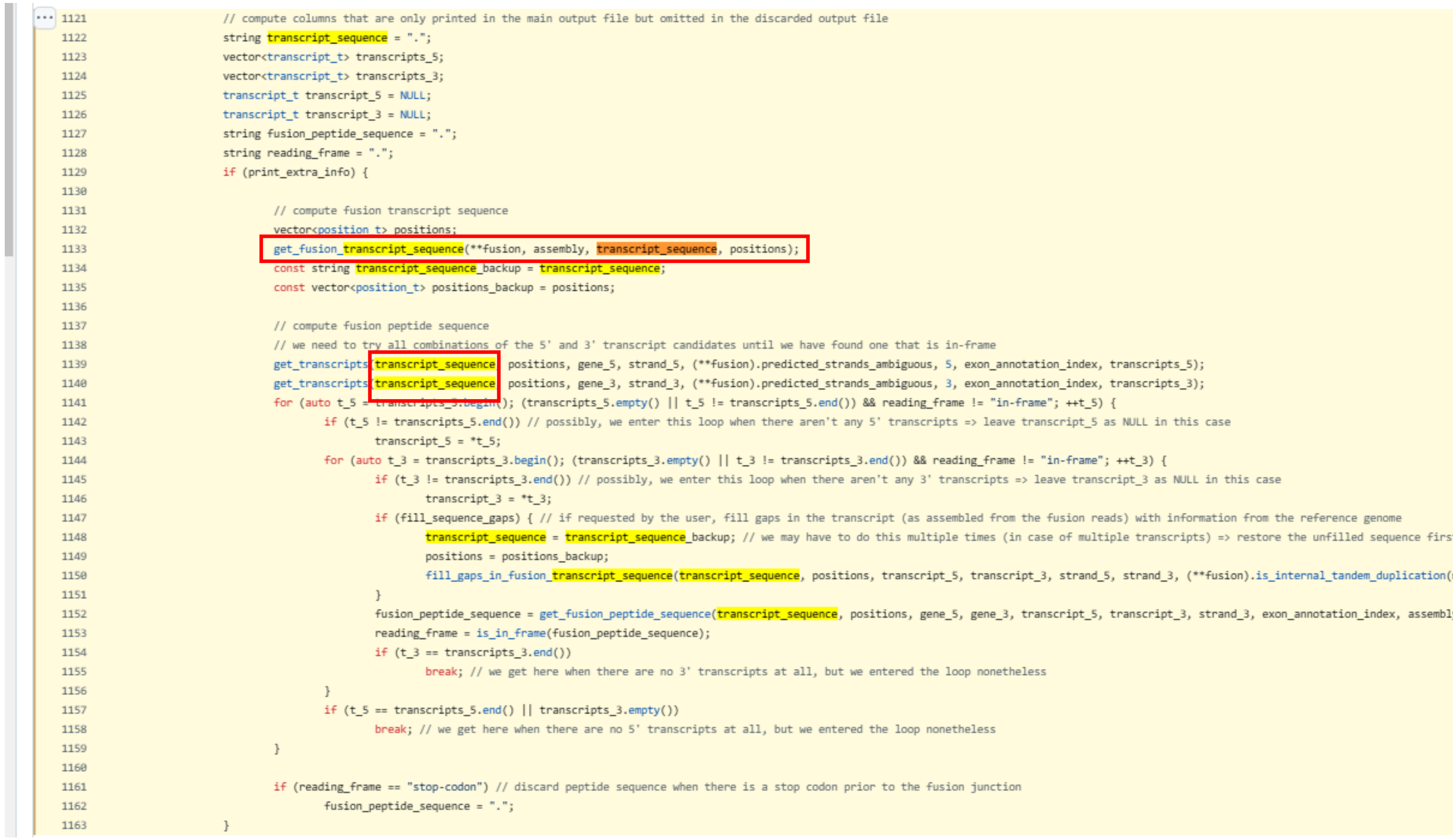
```
    // compute fusion peptide sequence
    // we need to try all combinations of the 5' and 3' transcript candidates until we have found one that is in-frame
    get_transcripts(transcript_sequence, positions, gene_5, strand_5, (**fusion).predicted_strands_ambiguous, 5, exon_annotation_index, transcripts_5);
    get_transcripts(transcript_sequence, positions, gene_3, strand_3, (**fusion).predicted_strands_ambiguous, 3, exon_annotation_index, transcripts_3);
    for (auto t_5 = transcripts_5.begin(); (transcripts_5.empty() || t_5 != transcripts_5.end()) && reading_frame != "in-frame"; ++t_5) {
        if (t_5 != transcripts_5.end()) // possibly, we enter this loop when there aren't any 5' transcripts => leave transcript_5 as NULL in this case
            transcript_5 = *t_5;
        for (auto t_3 = transcripts_3.begin(); (transcripts_3.empty() || t_3 != transcripts_3.end()) && reading_frame != "in-frame"; ++t_3) {
            if (t_3 != transcripts_3.end()) // possibly, we enter this loop when there aren't any 3' transcripts => leave transcript_3 as NULL in this case
                transcript_3 = *t_3;
            if (fill_sequence_gaps) { // if requested by the user, fill gaps in the transcript (as assembled from the fusion reads) with information from the
                transcript_sequence = transcript_sequence_backup; // we may have to do this multiple times (in case of multiple transcripts) => restore th
                positions = positions_backup;
                fill_gaps_in_fusion_transcript_sequence(transcript_sequence, positions, transcript_5, transcript_3, strand_5, strand_3, (**fusion).is_inte
            }
            fusion_peptide_sequence = get_fusion_peptide_sequence(transcript_sequence, positions, gene_5, gene_3, transcript_5, transcript_3, strand_3, exon_a
            reading_frame = is_in_frame(fusion_peptide_sequence);
            if (t_3 == transcripts_3.end())
                break; // we get here when there are no 3' transcripts at all, but we entered the loop nonetheless
        }
        if (t_5 == transcripts_5.end() || transcripts_3.empty())
            break; // we get here when there are no 5' transcripts at all, but we entered the loop nonetheless
    }

    if (reading_frame == "stop-codon") // discard peptide sequence when there is a stop codon prior to the fusion junction
        fusion_peptide_sequence = ".";
}
```

Arriba's translation

Ref:
- write_fusions_to_file
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/output_fusions.cpp#L1121-L1163
- get_fusion_transcript_sequence
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/output_fusions.cpp#L242C6-L466

- write_fusions_to_file (arriba.cpp)
  → main (arriba.cpp)
  → write_fusions_to_file (output_fusions.cpp)
  → get_transcripts (output_fusions.cpp)
  → transcript_sequence
  → get_fusion_transcript_sequence (output_fusions.cpp)

Note:
In this step,
The "get_fusion_transcript_sequence " piles up chimeric sequences next to the breakpoint and record them to "transcript_sequence" and "positions".



```
// compute columns that are only printed in the main output file but omitted in the discarded output file
string transcript_sequence = ".";
vector<transcript_t> transcripts_5;
vector<transcript_t> transcripts_3;
transcript_t transcript_5 = NULL;
transcript_t transcript_3 = NULL;
string fusion_peptide_sequence = ".";
string reading_frame = ".";
if (print_extra_info) {

    // compute fusion transcript sequence
    vector<position_t> positions;
    get_fusion_transcript_sequence(**fusion, assembly, transcript_sequence, positions);
    const string transcript_sequence_backup = transcript_sequence;
    const vector<position_t> positions_backup = positions;

    // compute fusion peptide sequence
    // we need to try all combinations of the 5' and 3' transcript candidates until we have found one that is in-frame
    get_transcripts(transcript_sequence, positions, gene_5, strand_5, (**fusion).predicted_strands_ambiguous, 5, exon_annotation_index, transcripts_5);
    get_transcripts(transcript_sequence, positions, gene_3, strand_3, (**fusion).predicted_strands_ambiguous, 3, exon_annotation_index, transcripts_3);
    for (auto t_5 = transcripts_5.begin(); (transcripts_5.empty() || t_5 != transcripts_5.end()) && reading_frame != "in-frame"; ++t_5) {
        if (t_5 != transcripts_5.end()) // possibly, we enter this loop when there aren't any 5' transcripts => leave transcript_5 as NULL in this case
            transcript_5 = *t_5;
        for (auto t_3 = transcripts_3.begin(); (transcripts_3.empty() || t_3 != transcripts_3.end()) && reading_frame != "in-frame"; ++t_3) {
            if (t_3 != transcripts_3.end()) // possibly, we enter this loop when there aren't any 3' transcripts => leave transcript_3 as NULL in this case
                transcript_3 = *t_3;
            if (fill_sequence_gaps) { // if requested by the user, fill gaps in the transcript (as assembled from the fusion reads) with information from the reference genome
                transcript_sequence = transcript_sequence_backup; // we may have to do this multiple times (in case of multiple transcripts) => restore the unfilled sequence firs
                positions = positions_backup;
                fill_gaps_in_fusion_transcript_sequence(transcript_sequence, positions, transcript_5, transcript_3, strand_5, strand_3, (**fusion).is_internal_tandem_duplication(
            }
            fusion_peptide_sequence = get_fusion_peptide_sequence(transcript_sequence, positions, gene_5, gene_3, transcript_5, transcript_3, strand_3, exon_annotation_index, assembl
            reading_frame = is_in_frame(fusion_peptide_sequence);
            if (t_3 == transcripts_3.end())
                break; // we get here when there are no 3' transcripts at all, but we entered the loop nonetheless
        }
        if (t_5 == transcripts_5.end() || transcripts_3.empty())
            break; // we get here when there are no 5' transcripts at all, but we entered the loop nonetheless
    }

    if (reading_frame == "stop-codon") // discard peptide sequence when there is a stop codon prior to the fusion junction
        fusion_peptide_sequence = ".";
}
```

Ref:
- get_transcripts
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/output_fusions.cpp#L719-L817

Arriba's translation

- get_transcripts (output_fusions.cpp)

Note:
For each breakpoint, the "main" function of the program arriba.cpp will first determine the "gene" and find a "transcript" whose exons match the splice pattern of the fusion transcript sequence well.

For each transcript,
function "get_transcripts" calculate score reflecting how well the transcribed bases match annotated exons.

The candidate transcripts annotation are retrieved from "exon_annotation_index" (input parameter).

Annotation algorithm:

For each in the fusion transcript within the sequence range (5'/3' sequence are process separately), "get_transcripts" function iterate through the overlapping exons from the "exon_annotation_index" and evaluates whether the base belongs to an exon of the current transcript
If so,
1. The score for the transcript is incremented.
2. If the breakpoint matches a splice site within 2 bases, an extra bonus score is added.
3. Coding region relevance: If the breakpoint occurs in the coding region of the transcript, this is flagged and may later influence transcript ranking

If non-matching bases identified, the score will be penalized.

Among all the evaluated transcripts,
the function will find the transcript with highest match quality score.
=>
If multiple transcripts have the same peak score the following criteria will be considered:
1. Is the breakpoint located in the coding region (is_coding_at_breakpoint)?
2. Number of transcribedUTR bases (transcribed_utr_bases).

If multiple transcripts identified with the same peak score,
the program preferred the one with smaller genomic ranges (smaller "size") and the lexicographical order of transcript IDs.

Remark: Only transcript candidate(s) with highest score will be stored to "best_transcripts" in the "get_transcripts " function.

=>
The function "write_fusions_to_file" will try all potential 5' + 3' combinations to find an in-frame candidate.

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.

Ref:

- read_chimeric_alignments
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/read_chimeric_alignments.cpp#L599-L753
- find_fusions
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/fusions.cpp#L203-L473

- External cpp library for bam file parsing
  https://broadinstitute.github.io/gamgee/doxygen/hts__memory_8h_source.html
  (bam1_t) bam_record
  https://broadinstitute.github.io/gamgee/doxygen/build_2contrib_2htslib_2src_2htslib_2htslib_2sam_8h_source.html
  (bam1_core_t) bam_record.core
  https://broadinstitute.github.io/gamgee/doxygen/structbam1__core__t.html
- fusion_t
  https://github.com/suhrig/arriba/blob/786f647a73b8ffb8bbe38b6607460c27e17e8846/source/common.hpp#L285

Arriba's break point detection

- read_chimeric_alignments (read_chimeric_alignments.cpp)
  → write_fusions_to_file (arriba.cpp)
  → read_chimeric_alignments (read_chimeric_alignments.cpp)

- find_fusions (fusions.cpp)
  → write_fusions_to_file (arriba.cpp)
  → find_fusions (fusions.cpp)

Note:
Arriba parses bam file (obtained from STAR aligner) to extract split-reads with SA tag and read-through alignments. (via function "read_chimeric_alignments")

The parsed alignments will then be processed by function "find_fusions" to classify reads.
⇒ Read level breakpoint identification (record in "fusions" (class fusions_t), arriba.cpp)

Summary:
⇒ Chimeric alignments and read-through alignments obtained from bam file are stored to "fusion_t" (breakpoint determination) ("read_chimeric_alignments")
⇒ Extract exon annotation form gtf file (Gene – potential exons) ("read_annotation_gtf")
⇒ Annotate the reads to exons according to the gtf annotation
(gene &/ exon annotation => exon annotation are conducted separately for each chimeric mate)
(class "gene_annotation_t")
⇒ Assign IDs (transcript ID) to genes ("write_fusions_to_file" → "get_transcripts")
⇒ Record fusions ("write_fusions_to_file")
  ⇒ Detect fusion events ("find_fusions")
  ⇒ Filter unwanted fusions (=> filter artifact, blacklist, …, etc.)
  ⇒ Transcript sequence ("write_fusions_to_file" → stored in string "transcript_sequence")

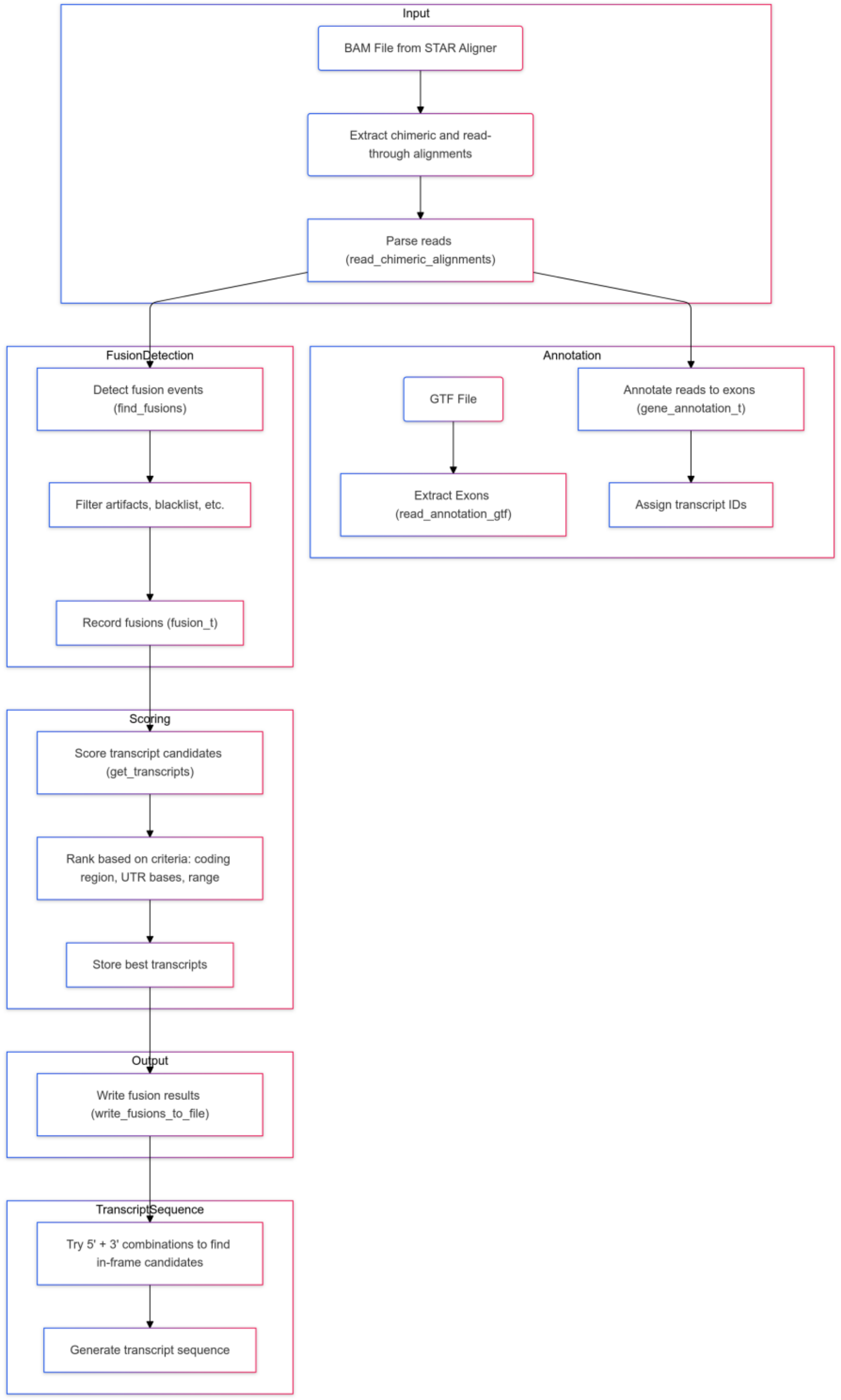
ACT GENOMICS ™

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.

Algorithm summary

- Algorithm summary plot (mermaid)

```
graph TD;

  subgraph Input
    BamFile("BAM File from STAR Aligner") --> ChimericReads("Extract chimeric and read-through alignments");
    ChimericReads --> ParseReads["Parse reads (read_chimeric_alignments)"];
  end;

  subgraph Annotation
    GTF("GTF File") --> ReadExons["Extract Exons (read_annotation_gtf)"];
    ParseReads --> Annotate["Annotate reads to exons (gene_annotation_t)"];
    Annotate --> AssignID["Assign transcript IDs"];
  end;

  subgraph FusionDetection
    ParseReads --> DetectFusions["Detect fusion events (find_fusions)"];
    DetectFusions --> FilterFusions["Filter artifacts, blacklist, etc."];
    FilterFusions --> RecordFusions["Record fusions (fusion_t)"];
  end;

  subgraph Scoring
    RecordFusions --> EvaluateTranscripts["Score transcript candidates (get_transcripts)"];
    EvaluateTranscripts --> RankTranscripts["Rank based on criteria: coding region, UTR bases, range"];
    RankTranscripts --> BestTranscripts["Store best transcripts"];
  end;

  subgraph Output
    BestTranscripts --> WriteFusions["Write fusion results (write_fusions_to_file)"];
  end;

  subgraph TranscriptSequence
    WriteFusions --> Combine5p3p["Try 5' + 3' combinations to find in-frame candidates"];
    Combine5p3p --> TranscriptSeq["Generate transcript sequence"];
  end;
```


Input
BAM File from STAR Aligner
Extract chimeric and read-through alignments
Parse reads (read_chimeric_alignments)
FusionDetection
Detect fusion events (find_fusions)
Filter artifacts, blacklist, etc.
Record fusions (fusion_t)
Annotation
GTF File
Extract Exons (read_annotation_gtf)
Annotate reads to exons (gene_annotation_t)
Assign transcript IDs
Scoring
Score transcript candidates (get_transcripts)
Rank based on criteria: coding region, UTR bases, range
Store best transcripts
Output
Write fusion results (write_fusions_to_file)
TranscriptSequence
Try 5' + 3' combinations to find in-frame candidates
Generate transcript sequence

ACT GENOMICS™

© 2025 ACT Genomics CO., LTD. Proprietary and Confide

Splicing variants

- fusion.discarded.tsv (no target variant)

- Remark:
  Arriba‘s false positive events:

  read-through/deletion
  (Deletions with a size in the range of introns (<400kb);
  contains all but the last exon of the 5’ gene and all but the first exon of the 3’ gene)

Ref. command:

```
more
/mnt/RD_Develop/sandyteng/FusionCaptureTools/testresult/arriba_v2.
4.0_grch38/fusions.discarded.tsv | awk -F"\t" '{print""$1"-
"$2";(chr"$5",chr"$6")\t"$9"\t"$1"("$23")\t"$2"("$24")\t"$5"\t"$6"\t"$1
0"\t"$11"\t"$12"\t"$27"\t"$13"\t"$14"\t"$15""}' | grep "MET-MET“
```

```
# MET-MET => target  “MET-MET;(chr7:116771654,chr7:116774881)”
MET-MET;(chr7:116771617,chr7:116771619) deletion/read-through   MET(.)  MET(.)  7:116771617    7:116771619    0    0    0    read_through(2) 13374  13564  low
MET-MET;(chr7:116774901,chr7:116771564) duplication    MET(.)  MET(.)  7:116774901    7:116771564    0    0    0    small_insert_size    12664  13374  low
MET-MET;(chr7:116771606,chr7:116771613) deletion/read-through   MET(.)  MET(.)  7:116771606    7:116771613    0    0    0    duplicates(2),read_through(1)   13374  13564  low

# EGFR-EGFR => target “EGFR-EGFR;(chr7:55019365,chr7:55155830)”
EGFR-EGFR;(chr7:55151335,chr7:55146672) duplication    EGFR(.) EGFR(.) 7:55151335    7:55146672    0    0    0    small_insert_size    10    6    low
EGFR-EGFR;(chr7:55205337,chr7:55202591) duplication    EGFR(.) EGFR(.) 7:55205337    7:55202591    0    0    0    small_insert_size    5055   4895   low
EGFR-EGFR;(chr7:55101372,chr7:55174759) deletion/read-through   EGFR(.) EGFR(.) 7:55101372    7:55174759    0    0    0    duplicates(2),read_through(1)   3    22    low
EGFR-EGFR;(chr7:55200388,chr7:55198814) duplication    EGFR(.) EGFR(.) 7:55200388    7:55198814    0    0    0    small_insert_size    1966   1620   low
EGFR-EGFR;(chr7:55200384,chr7:55198818) duplication    EGFR(.) EGFR(.) 7:55200384    7:55198818    0    0    0    small_insert_size    1966   1620   low
EGFR-EGFR;(chr7:55204737,chr7:55204651) duplication/ITD EGFR(.) EGFR(.) 7:55204737    7:55204651    0    0    0    low_entropy(1)  10    1    low
EGFR-EGFR;(chr7:55205357,chr7:55202607) duplication    EGFR(.) EGFR(.) 7:55205357    7:55202607    0    0    0    small_insert_size    4346   63    low
EGFR-EGFR;(chr7:55205354,chr7:55202607) duplication    EGFR(.) EGFR(.) 7:55205354    7:55202607    0    0    0    small_insert_size    4346   63    low
EGFR-EGFR;(chr7:55200409,chr7:55173918) duplication    EGFR(.) EGFR(.) 7:55200409    7:55173918    0    0    0    duplicates(1)   1367   230   low
EGFR-EGFR;(chr7:55200383,chr7:55198818) duplication    EGFR(.) EGFR(.) 7:55200383    7:55198818    0    0    0    small_insert_size    1966   1620   low
EGFR-EGFR;(chr7:55143436,chr7:55143448) deletion/read-through   EGFR(.) EGFR(.) 7:55143436    7:55143448    0    0    0    read_through(1) 41    55    low
EGFR-EGFR;(chr7:55095803,chr7:55095797) duplication/ITD EGFR(.) EGFR(.) 7:55095803    7:55095797    0    0    0    low_entropy(1)  41    42    low
EGFR-EGFR;(chr7:55095896,chr7:55095813) duplication/ITD EGFR(.) EGFR(.) 7:55095896    7:55095813    0    0    0    hairpin(1)    53    54    low
EGFR-EGFR;(chr7:55205328,chr7:55202584) duplication    EGFR(.) EGFR(.) 7:55205328    7:55202584    0    0    0    small_insert_size    5055   4895   low

# AR-AR => target AR-V7 “AR-AR;(chrX:67643256,chrX:67696075)”
(potential AR:2-AR:4)
AR-AR;(chrX:67694692,chrX:67643355)    duplication    AR(.)   AR(.)   X:67694692    X:67643355    1    0    0    duplicates(2),intragenic_exonic 11496  0    low
(potential AR:intron-AR:intron)
AR-AR;(chrX:67653778,chrX:67653898)    duplication    AR(.)   AR(.)   X:67653778    X:67653898    0    0    0    small_insert_size    1    1    low
```

```
Grch 38 exons (exon chromosome start end)
>ENST00000504326.5    ENSG00000169083.18    AR    NM_001348061.1  chrX    67544820    67696075    +    4    151256  3645
1    chrX    67544820    67546762    5    5'UTR
2    chrX    67643256    67643407    0
3    chrX    67686010    67686126    0
4    chrX    67694673    67696075    3    3'UTR

>ENST00000275493.7    ENSG00000146648.20    EGFR    NM_005228.5    chr7    55019017    55211628    +    28    192612  10175
1    chr7    55019017    55019365    5    5'UTR
2    chr7    55142286    55142437    0
3    chr7    55143305    55143488    0
4    chr7    55146606    55146740    0
5    chr7    55151294    55151362    0
6    chr7    55152546    55152664    0
7    chr7    55154011    55154152    0
8    chr7    55155830    55155946    0
>ENST00000397752.8    ENSG00000105976.16    MET    NM_000245.4    chr7    116672196    116798377    +    21    126182  7022
13    chr7    116771498    116771654    0
14    chr7    116771849    116771989    0
15    chr7    116774881    116775111    0
```

```
python rmats.py --s1
/mnt/RD_Develop/sandyteng/SplicingVariantCaptureTools/rmats_v4.3.0/AANB02_184_IDD705504_IVTALL-1.s1.file.txt --gtf
/mnt/RD_Develop/sandyteng/FusionCaptureTools/refdb_arriba/RefSeq_hg38.gtf --bi
/mnt/RD_Develop/sandyteng/FusionCaptureTools/refdb_arriba/STAR_index_GRCh38_RefSeq_hg38/ -t paired --readLength 50 -
-nthread 4 --od
/mnt/RD_Develop/sandyteng/SplicingVariantCaptureTools/testresult/rmats_v4.3.0/IVTALL_AANB02_184_IDD705504/output/ --
tmp
/mnt/RD_Develop/sandyteng/SplicingVariantCaptureTools/testresult/rmats_v4.3.0/IVTALL_AANB02_184_IDD705504/tmp_outp
ut_post/
```

Arriba versus rMATs

- STAR parameter comparison
  - Arriba (2.4.0) => favor fusions
  - rMATs (rmats:v4.3.0) => favor splicing variants
    => no splicing breakpoints identified (cannot report the 5 splicing variants within the IVTALL sample (amplicon-based data obtained from v4 assay))

Feature	Arriba Command	rMATS Command
Purpose	Fusion detection (Arriba).	Splicing analysis (rMATS).
Alignment Type	Relaxed.	End-to-End.
Output	BAM Unsorted (unmapped included).	BAM SortedByCoordinate.
Chimeric Reads	Chimeric segments for fusion detection.	Minimal chimeric settings.
GTF Annotations	Not required.	Uses GTF for splicing analysis.
Mismatch Allowance	Relaxed (--outFilterMultimapNmax 50).	Strict (--outFilterMismatchNmax 3).
Spliced Settings	Optimized for chimeric junctions.	Optimized for spliced junctions.

Ref:
- Test image: (on terminal 177)
- Reference issue: https://actg.atlassian.net/browse/ABIE-947

ACT GENOMICS ™

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.

MET:13-MET:15

```
Grch 38 exons (exon chromosome start end)
>ENST00000397752.8    ENSG00000105976.16    MET    NM_000245.4    chr7   116672196    116798377    +    21    126182  7022
13    chr7   116771498    116771654    0
14    chr7   116771849    116771989    0
15    chr7   116774881    116775111    0
```

Ref:

```
samtools view Aligned.sortedByCoord.out.bam  | grep "NB552518:184:HGFWYAFX7:1:21211:7523:12923“
```

Remark:

"NB552518:184:HGFWYAFX7:1:21211:7523:12923“ → MET:13-MET:15

- Arriba’s (STAR) aligned.bam

# Aligned.sortedByCoord.out.bam

Exon 13   Exon 15

```
NB552518:184:HGFWYAFX7:1:21211:7523:12923     99    7  116411619  1     24S90M3226N37M =      116411619     3374
GCGTGAATGTAAGCGTGACGGGGAGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCG
AAAAAEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEAEEEEA<AAAA/EAEEEA//AEA/<EAAAE<EEEAAAA/EEEEAAEEE/A<EEEEAAEEAEAE/ NH:i:3  HI:i:1  AS:i:0  nM:i:0  NM:i:0      SA:Z:X,18911818,-,127H14M10S,1,0;
NB552518:184:HGFWYAFX7:1:21211:7523:12923     355   7  116411619  1     24S90M3226N37M =      116411619     3374
GCGTGAATGTAAGCGTGACGGGGAGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCG
AAAAAEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEAEEEEA<AAAA/EAEEEA//AEA/<EAAAE<EEEAAAA/EEEEAAEEE/A<EEEEAAEEAEAE/ NH:i:3  HI:i:2  AS:i:0  nM:i:0  NM:i:0      SA:Z:X,97674443,+,10S14M127H,1,0;
NB552518:184:HGFWYAFX7:1:21211:7523:12923     355   7  116411619  1     24S90M3226N37M =      116411619     3374
GCGTGAATGTAAGCGTGACGGGGAGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCG
AAAAAEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEAEEEEA<AAAA/EAEEEA//AEA/<EAAAE<EEEAAAA/EEEEAAEEE/A<EEEEAAEEAEAE/ NH:i:3  HI:i:3  AS:i:0  nM:i:0  NM:i:0      SA:Z:17,850483,+,10S14M127H,1,0;
NB552518:184:HGFWYAFX7:1:21211:7523:12923     147    7  116411619  1     3S90M3226N58M  =      116411619     -3374
GGAGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCGACAAGTGCAGTATCCTCTGAC
<EEEAAEE/EAEEEEAEE<EEA<EAE/E<EEEEAEEEEEEAAEEEEEEEEEEEEEE/EAAEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEAEEEEEEEEEEEEEEE6/EEEEEEEEEEEEEEEEEEEEAAAAA NH:i:3  HI:i:1  AS:i:0  nM:i:0  NM:i:0
NB552518:184:HGFWYAFX7:1:21211:7523:12923     403    7  116411619  1     3S90M3226N58M  =      116411619     -3374
GGAGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCGACAAGTGCAGTATCCTCTGAC
<EEEAAEE/EAEEEEAEE<EEA<EAE/E<EEEEAEEEEEEAAEEEEEEEEEEEEEE/EAAEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEAEEEEEEEEEEEEEEE6/EEEEEEEEEEEEEEEEEEEEAAAAA NH:i:3  HI:i:2  AS:i:0  nM:i:0  NM:i:0
NB552518:184:HGFWYAFX7:1:21211:7523:12923     403    7  116411619  1     3S90M3226N58M  =      116411619     -3374
GGAGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCGACAAGTGCAGTATCCTCTGAC
<EEEAAEE/EAEEEEAEE<EEA<EAE/E<EEEEAEEEEEEAAEEEEEEEEEEEEEE/EAAEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEAEEEEEEEEEEEEEEE6/EEEEEEEEEEEEEEEEEEEEAAAAA NH:i:3  HI:i:3  AS:i:0  nM:i:0  NM:i:0
NB552518:184:HGFWYAFX7:1:21211:7523:12923     2401   17  850483     1     10S14M127H     7      116411619     0      GCGTGAATGTAAGCGTGACGGGGA       AAAAAEEEEEEEEEEEEEEEEEEA       NH:i:3  HI:i:3  AS:i:0  nM:i:0  NM:i:0  SA:Z:7,116411619,+,24S90M3226N37M,1,0;
NB552518:184:HGFWYAFX7:1:21211:7523:12923     2161   X  18911818   1     127H14M10S     7      116411619     0      TCCCCGTCACGCTTACATTCACGC       AEEEEEEEEEEEEEEEEEEAAAAA       NH:i:3  HI:i:1  AS:i:0  nM:i:0  NM:i:0  SA:Z:7,116411619,+,24S90M3226N37M,1,0;
NB552518:184:HGFWYAFX7:1:21211:7523:12923     2401   X  97674443   1     10S14M127H     7      116411619     0      GCGTGAATGTAAGCGTGACGGGGA       AAAAAEEEEEEEEEEEEEEEEEEA       NH:i:3  HI:i:2  AS:i:0  nM:i:0  NM:i:0  SA:Z:7,116411619,+,24S90M3226N37M,1,0;

NB552518:184:HGFWYAFX7:4:11409:22009:4038     99    7  116411618  255    21S91M3226N39M =       116411618     3375
GCGTGAATGTAAGCGTGACGGGGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCGAC
AAAAAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEE/EEEEEAEEEEEEEEEEEEEEEAEEEEEE///<<</E/AEEE/A<EA/AAA/AA/AEEAAEE/EEAEEAE<EAAAEEEEEEEAAAAEAE/</ NH:i:1  HI:i:1  AS:i:280      nM:i:0      NM:i:0
NB552518:184:HGFWYAFX7:4:11409:22009:4038     147   7  116411618  255    2S91M3226N58M  =       116411618     -3375
GGGGATTGATTGCTGGTGTTGTCTCAATATCAACAGCACTGTTATTACTACTTGGGTTTTTCCTGTGGCTGAAAAAGAGAAAGCAAATTAAAGATCAGTTTCCTAATTCATCTCAGAACGGTTCATGCCGACAAGTGCAGTATCCTCTGAC
<EEEAAEEEEEEEEEE<EEEEEEEEEEEAEEEAEEEEEEEEEEEEAAEEEEEEEEEEE6EAEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEE/EEEEEEEEEEEEEEEEEEEEAAAAA NH:i:1  HI:i:1  AS:i:280      nM:i:0      NM:i:0
```

ACT GENOMICS ™

ACT GENOMICS ™

Make Personalized Medicine Accessible to All

© 2025 ACT Genomics CO., LTD. Proprietary and Confidential. All Rights Reserved.