REAT

Release 0.6.1

Luis Yanes

Mar 08, 2022

CONTENTS:

REAT is a robust easy-to-use genome annotation toolkit for turning high-quality genome assemblies into usable and informative resources. REAT makes use of state-of-the-art annotation tools and is robust to varying quality and sources of molecular evidence.

REAT provides an integrated environment that comprises both a set of workflows geared towards integrating multiple sources of evidence into a genome annotation, and an execution environment for these workflows.

CHAPTER

ONE

INSTALLATION

To install REAT you can:

```
git clone https://github.com/ei-corebioinformatics/reat
wget https://github.com/broadinstitute/cromwell/releases/download/62/cromwell-62.jar
conda env create -f reat/reat.yml
pip install ./reat
```

These commands will download the cromwell binary required to execute the workflows and make REAT available in the 'reat' conda environment which can be activated using:

```
conda activate reat
```

Each task in the workflow is configured with default resource requirements appropriate for most tasks, but these can be overriden by user provided ones. For examples of resource configuration files, refer to each module's description.

To configure the cromwell engine, there are two relevant files, the cromwell runtime options and the workflow options files.

The cromwell engine can be configured to run in your environment using a file such as:

```
include required(classpath("application"))

database {
    profile = "slick.jdbc.HsqldbProfile$"
    db {
        driver = "org.hsqldb.jdbcDriver"
        url = """
        jdbc:hsqldb:file:cromwell-executions/cromwell-db/cromwell-db;
        shutdown=false;
        hsqldb.default_table_type=cached;hsqldb.tx=mvcc;
        hsqldb.result_max_memory_rows=10000;
        hsqldb.large_data=true;
        hsqldb.applog=1;
        hsqldb.lob_compressed=true;
        hsqldb.script_format=3
        """
        connectionTimeout = 120000
        numThreads = 1
        }
}

# concurrent-job-limit = 2
# max-concurrent-workflows = 1
```

```
akka.http.server.request-timeout = 30s

call-caching {
  # Allows re-use of existing results for jobs you've already run
  # (default: false)
  enabled = true

  # Whether to invalidate a cache result forever if we cannot reuse them. Disable this␣
↪if you expect some cache copies
  # to fail for external reasons which should not invalidate the cache (e.g. auth␣
↪differences between users):
  # (default: true)
  invalidate-bad-cache-results = true
}

backend {
  default = slurm
  providers {
    slurm {
      actor-factory = "cromwell.backend.impl.sfs.config.ConfigBackendLifecycleActorFactory"
      config {
        concurrent-job-limit = 50

        filesystems {
          local {
            localization: [
              # for local SLURM, hardlink doesn't work. Options for this and caching: ,␣
↪"soft-link" , "hard-link", "copy"
              "soft-link", "copy"
            ]
            ## call caching config relating to the filesystem side
            caching {
              # When copying a cached result, what type of file duplication should occur.
↪ Attempted in the order listed below:
              duplication-strategy: [
                "soft-link"
              ]
              hashing-strategy: "path"
              # Possible values: file, path, path+modtime
              # "file" will compute an md5 hash of the file content.
              # "path" will compute an md5 hash of the file path. This strategy will␣
↪only be effective if the duplication-strategy (above) is set to "soft-link",
              # in order to allow for the original file path to be hashed.

              check-sibling-md5: false
              # When true, will check if a sibling file with the same name and the .md5␣
↪extension exists, and if it does, use the content of this file as a hash.
              # If false or the md5 does not exist, will proceed with the above-defined␣
↪hashing strategy.
            }
          }
        }

        runtime-attributes = """
```

```
        Int runtime_minutes = 1440
        Int cpu = 4
        Int memory_mb = 8000
        String? constraints
        String? queue = "ei-medium"
        """

        submit = """
        if [ "" == "${queue}" ]
        then
                sbatch -J ${job_name} --constraint="${constraints}" -D ${cwd} -o ${out}␣
↪-e ${err} -t ${runtime_minutes} \
                -p ei-medium \
            ${"-c " + cpu} \
            --mem ${memory_mb} \
            --wrap "/bin/bash
            ${script}"
        else
                sbatch -J ${job_name} --constraint="${constraints}" -D ${cwd} -o ${out}␣
↪-e ${err} -t ${runtime_minutes} \
                -p ${queue} \
            ${"-c " + cpu} \
            --mem ${memory_mb} \
            --wrap "/bin/bash
            ${script}"
        fi

        """
        kill = "scancel ${job_id}"
        check-alive = "squeue -j ${job_id}"
        job-id-regex = "Submitted batch job (\\d+).*"
        exit-code-timeout-seconds = 45
      }
    }
  }
}
```

The workflow options can be used to activate the caching behaviour in cromwell, i.e:

```
{
    "write_to_cache": true,
    "read_from_cache": true,
    "memory_retry_multiplier" : 1.5
}
```

1.1 Transcriptome Workflow

The intention of the transcriptome workflow is to use a variety of data types, from short reads to long reads of varied quality and length.

The data input for the workflow can be defined through the use of comma separated files one for short read samples and another for long read samples. These samples are then processed in several steps, first they are aligned to the genome, then assembled into transcripts, junctions are determined from the data and finally they are combined into a consolidated set of gene models.

The aligner and assembly programs used for short and long read samples can be selected through command line arguments. There are also command line arguments to select extra options to be applied at each step.

In case an annotation is available, this can be provided for junctions and reference models to be extracted and these can then be augmented using the evidence present in the data.

```
Welcome to REAT
version - 0.6.1

Command-line call:
/home/docs/checkouts/readthedocs.org/user_builds/reat/envs/stable/bin/reat transcriptome␣
↪--help


usage: reat transcriptome [-h] --reference REFERENCE
                          [--samples SAMPLES [SAMPLES ...]]
                          [--csv_paired_samples CSV_PAIRED_SAMPLES]
                          [--csv_long_samples CSV_LONG_SAMPLES]
                          [--annotation ANNOTATION]
                          [--annotation_score ANNOTATION_SCORE]
                          [--check_reference]
                          [--mode {basic,update,only_update}]
                          [--extra_junctions EXTRA_JUNCTIONS]
                          [--skip_mikado_long] [--filter_HQ_assemblies]
                          [--filter_LQ_assemblies]
                          [--parameters_file PARAMETERS_FILE]
                          [--genetic_code GENETIC_CODE]
                          [--all_extra_config ALL_EXTRA_CONFIG]
                          [--long_extra_config LONG_EXTRA_CONFIG]
                          [--lq_extra_config LQ_EXTRA_CONFIG]
                          --all_scoring_file ALL_SCORING_FILE
                          [--long_scoring_file LONG_SCORING_FILE]
                          [--long_lq_scoring_file LONG_LQ_SCORING_FILE]
                          [--homology_proteins HOMOLOGY_PROTEINS]
                          [--separate_mikado_LQ SEPARATE_MIKADO_LQ]
                          [--exclude_LQ_junctions]
                          [--short_reads_aligner {hisat,star}]
                          [--skip_2pass_alignment]
                          [--HQ_aligner {minimap2,gmap,2pass,2pass_merged}]
                          [--LQ_aligner {minimap2,gmap,2pass,2pass_merged}]
                          [--min_identity [0-100]]
                          [--min_intron_len MIN_INTRON_LEN]
                          [--max_intron_len MAX_INTRON_LEN]
                          [--max_intron_len_ends MAX_INTRON_LEN_ENDS]
                          [--PR_hisat_extra_parameters PR_HISAT_EXTRA_PARAMETERS]
```

```
                        [--PR_star_extra_parameters PR_STAR_EXTRA_PARAMETERS]
                        [--HQ_aligner_extra_parameters HQ_ALIGNER_EXTRA_PARAMETERS]
                        [--LQ_aligner_extra_parameters LQ_ALIGNER_EXTRA_PARAMETERS]
                        [--skip_scallop]
                        [--HQ_assembler {filter,merge,stringtie,stringtie_collapse}]
                        [--LQ_assembler {filter,merge,stringtie,stringtie_collapse}]
                        [--HQ_min_identity [0-100]]
                        [--HQ_min_coverage [0-100]]
                        [--HQ_assembler_extra_parameters HQ_ASSEMBLER_EXTRA_PARAMETERS]
                        [--LQ_min_identity [0-100]]
                        [--LQ_min_coverage [0-100]]
                        [--LQ_assembler_extra_parameters LQ_ASSEMBLER_EXTRA_PARAMETERS]
                        [--PR_stringtie_extra_parameters PR_STRINGTIE_EXTRA_PARAMETERS]
                        [--PR_scallop_extra_parameters PR_SCALLOP_EXTRA_PARAMETERS]
                        [--extra_parameters EXTRA_PARAMETERS]
                        [--orf_caller {prodigal,transdecoder,none}]
                        [--orf_calling_proteins ORF_CALLING_PROTEINS]

optional arguments:
  -h, --help            show this help message and exit
  --reference REFERENCE
                        Reference FASTA to annotate (default: None)
  --samples SAMPLES [SAMPLES ...]
                        Reads organised in the input specification for REAT, for more␣
↪information please look at https://github.com/ei-corebioinformatics/reat
                        for an example (default: None)
  --csv_paired_samples CSV_PAIRED_SAMPLES
                        CSV formatted input paired read samples. Without headers.

                        The CSV fields are as follows name, strand, files (because this␣
↪is an array that can contain one or more pairs,
                        this fields' values are separated by semi-colon and space. Files␣
↪in a pair are separated by semi-colon pairs are
                        separated by a single space), merge, score, is_ref, exclude_
↪redundant.

                        sample_strand takes values 'fr-firststrand', 'fr-unstranded',␣
↪'fr-secondstrand'

                        merge, is_ref and exclude_redundant are boolean and take values␣
↪'true', 'false'

                        Example:
                        PR1,fr-secondstrand,A_R1.fq;A_R2.fq /samples/paired/B1.fq;/
↪samples/paired/B2.fq,false,2
                         (default: None)
  --csv_long_samples CSV_LONG_SAMPLES
                        CSV formatted input long read samples. Without headers."
                        The CSV fields are as follows name, strand, files (space␣
↪separated if there is more than one), quality, score, is_ref, exclude_redundant

                        sample_strand takes values 'fr-firststrand', 'fr-unstranded',␣
↪'fr-secondstrand'
                        quality takes values 'low', 'high'
```

```
                        is_ref and exclude_redundant are booleans and take values 'true',
↪ 'false'

                        Example:

                        Sample1,fr-firststrand,A.fq /samples/long/B.fq ./inputs/C.fq,low,
↪2 (default: None)
  --annotation ANNOTATION
                        Annotation of the reference, this file will be used as the base␣
↪for the new annotation which will incorporate from the
                        available evidence new gene models or update existing ones␣
↪(default: None)
  --annotation_score ANNOTATION_SCORE
                        Score for models in the reference annotation file (default: 1)
  --check_reference     At mikado stage, annotation models will be evaluated in the same␣
↪manner as RNA-seq based models, removing any models
                        deemed incorrect (default: False)
  --mode {basic,update,only_update}
                        basic: Annotation models are treated the same as the RNA-Seq␣
↪models at the pick stage.
                        update: Annotation models are prioritised but also novel loci␣
↪are reported.
                        only_update: Annotation models are prioritised and non-reference␣
↪loci are excluded. (default: basic)
  --extra_junctions EXTRA_JUNCTIONS
                        Extra junctions provided by the user, this file will be used as␣
↪a set of valid junctions for alignment of short and
                        long read samples, in the case of long reads, these junctions␣
↪are combined with the results of portcullis whenever
                         short read samples have been provided as part of the input␣
↪datasets (default: None)
  --skip_mikado_long    Disables generation of the long read only mikado run (default:␣
↪False)
  --filter_HQ_assemblies
                        Use all the junctions available to filter the HQ_assemblies␣
↪before mikado (default: False)
  --filter_LQ_assemblies
                        Use all the junctions available to filter the LQ_assemblies␣
↪before mikado (default: False)
  --parameters_file PARAMETERS_FILE
                        Base parameters file, this file can be the output of a previous␣
↪REAT run which will be used as the base for a new
                        parameters file written to the output_parameters_file argument␣
↪(default: None)
  --genetic_code GENETIC_CODE
                        Parameter for the translation table used in Mikado for␣
↪translating CDS sequences, and for ORF calling, can
                        take values in the genetic code range of NCBI as an integer. E.g␣
↪1, 6, 10 or when using TransDecoder as ORF
                        caller, one of: Universal, Tetrahymena, Acetabularia, Ciliate,␣
↪Dasycladacean, Hexamita, Candida, Euplotid,
                        SR1_Gracilibacteria, Pachysolen_tannophilus, Peritrich. 0 is␣
↪equivalent to Standard, NCBI #1, but only ATG is
                        considered a valid start codon. (default: 0)
```

```
Mikado:
  Parameters for Mikado runs

  --all_extra_config ALL_EXTRA_CONFIG
                        External configuration file for Paired and Long reads mikado␣
↪(default: None)
  --long_extra_config LONG_EXTRA_CONFIG
                        External configuration file for Long reads mikado run (default:␣
↪None)
  --lq_extra_config LQ_EXTRA_CONFIG
                        External configuration file for Low-quality long reads only␣
↪mikado run (this is only applied when
                        'separate_mikado_LQ' is enabled) (default: None)
  --all_scoring_file ALL_SCORING_FILE
                        Mikado long and short scoring file (default: None)
  --long_scoring_file LONG_SCORING_FILE
                        Mikado long scoring file (default: None)
  --long_lq_scoring_file LONG_LQ_SCORING_FILE
                        Mikado low-quality long scoring file (default: None)
  --homology_proteins HOMOLOGY_PROTEINS
                        Homology proteins database, used to score transcripts by Mikado␣
↪(default: None)
  --separate_mikado_LQ SEPARATE_MIKADO_LQ
                        Specify whether or not to analyse low-quality long reads␣
↪separately from high-quality, this option generates an
                        extra set of mikado analyses including low-quality data␣
↪(default: None)
  --exclude_LQ_junctions
                        When this parameter is defined, junctions derived from␣
↪low-quality long reads will not be included in the set of
                        valid junctions for the mikado analyses (default: False)

Alignment:
  Parameters for alignment of short and long reads

  --short_reads_aligner {hisat,star}
                        Choice of short read aligner (default: hisat)
  --skip_2pass_alignment
                        If not required, the second round of alignments for 2passtools␣
↪can be skipped when this parameter
                        is active (default: False)
  --HQ_aligner {minimap2,gmap,2pass,2pass_merged}
                        Choice of aligner for high-quality long reads (default: minimap2)
  --LQ_aligner {minimap2,gmap,2pass,2pass_merged}
                        Choice of aligner for low-quality long reads (default: minimap2)
  --min_identity [0-100]
                        Minimum alignment identity (passed only to gmap) (default: 90)
  --min_intron_len MIN_INTRON_LEN
                        Where available, the minimum intron length allowed will be␣
↪specified for the aligners (default: 20)
  --max_intron_len MAX_INTRON_LEN
                        Where available, the maximum intron length allowed will be␣
↪specified for the aligners (default: 200000)
```

```
  --max_intron_len_ends MAX_INTRON_LEN_ENDS
                        Where available, the maximum *boundary* intron length allowed␣
↪will be specified for the aligner, when specified
                        this implies max_intron_len only applies to the *internal*␣
↪introns and this parameter to the *boundary* introns (default: 100000)
  --PR_hisat_extra_parameters PR_HISAT_EXTRA_PARAMETERS
                        Extra command-line parameters for the selected short read␣
↪aligner, please note that extra parameters are not
                        validated and will have to match the parameters available for␣
↪the selected read aligner (default: None)
  --PR_star_extra_parameters PR_STAR_EXTRA_PARAMETERS
                        Extra command-line parameters for the selected short read␣
↪aligner, please note that extra parameters are not
                        validated and will have to match the parameters available for␣
↪the selected read aligner (default: None)
  --HQ_aligner_extra_parameters HQ_ALIGNER_EXTRA_PARAMETERS
                        Extra command-line parameters for the selected long read aligner,
↪ please note that extra parameters are not
                        validated and will have to match the parameters available for␣
↪the selected read aligner (default: None)
  --LQ_aligner_extra_parameters LQ_ALIGNER_EXTRA_PARAMETERS
                        Extra command-line parameters for the selected long read aligner,
↪ please note that extra parameters are not
                        validated and will have to match the parameters available for␣
↪the selected read aligner (default: None)

Assembly:
  Parameters for assembly of short and long reads

  --skip_scallop
  --HQ_assembler {filter,merge,stringtie,stringtie_collapse}
                        Choice of long read assembler.
                        - filter: Simply filters the reads based on identity and coverage
                        - merge: cluster the input transcripts into loci, discarding␣
↪"duplicated" transcripts (those with the same exact
                                introns and fully contained or equal boundaries). This␣
↪option also discards contained transcripts
                        - stringtie: Assembles the long reads alignments into transcripts
                        - stringtie_collapse: Cleans and collapses long reads but does␣
↪not assemble them (default: filter)
  --LQ_assembler {filter,merge,stringtie,stringtie_collapse}
                        Choice of long read assembler.
                        - filter: Simply filters the reads based on identity and coverage
                        - merge: cluster the input transcripts into loci, discarding␣
↪"duplicated" transcripts (those with the same exact
                                introns and fully contained or equal boundaries). This␣
↪option also discards contained transcripts
                        - stringtie: Assembles the long reads alignments into transcripts
                        - stringtie_collapse: Cleans and collapses long reads but does␣
↪not assembles them (default: stringtie_collapse)
  --HQ_min_identity [0-100]
                        When the 'filter' option is selected, this parameter defines the␣
↪minimum identity used to filtering (default: None)
  --HQ_min_coverage [0-100]
```

```
                        When the 'filter' option is selected, this parameter defines the␣
↪minimum coverage used for filtering (default: None)
  --HQ_assembler_extra_parameters HQ_ASSEMBLER_EXTRA_PARAMETERS
                        Extra parameters for the long reads assembler, please note that␣
↪extra parameters are not validated and will have to
                        match the parameters available for the selected assembler␣
↪(default: None)
  --LQ_min_identity [0-100]
                        When the 'filter' option is selected, this parameter defines the␣
↪minimum identity used to filtering (default: None)
  --LQ_min_coverage [0-100]
                        When the 'filter' option is selected, this parameter defines the␣
↪minimum coverage used for filtering (default: None)
  --LQ_assembler_extra_parameters LQ_ASSEMBLER_EXTRA_PARAMETERS
                        Extra parameters for the long reads assembler, please note that␣
↪extra parameters are not validated and will have to
                        match the parameters available for the selected assembler␣
↪(default: None)
  --PR_stringtie_extra_parameters PR_STRINGTIE_EXTRA_PARAMETERS
                        Extra parameters for stringtie, please note that extra␣
↪parameters are not validated and will have to
                        match the parameters available for stringtie (default: None)
  --PR_scallop_extra_parameters PR_SCALLOP_EXTRA_PARAMETERS
                        Extra parameters for scallop, please note that extra parameters␣
↪are not validated and will have to
                        match the parameters available for scallop (default: None)

Portcullis:
  Parameters specific to portcullis

  --extra_parameters EXTRA_PARAMETERS
                        Extra parameters for portcullis execution (default: None)

ORF Caller:
  Parameters for ORF calling programs

  --orf_caller {prodigal,transdecoder,none}
                        Choice of available orf calling softwares (default: prodigal)
  --orf_calling_proteins ORF_CALLING_PROTEINS
                        Set of proteins to be aligned to the genome for orf prediction␣
↪by Transdecoder (default: None)
```

1.1.1 Sample files

The way samples are organised in the input files reflects how the files that correspond to the sample will be processed. Data can be combined or kept separate at different stages of the workflow in accordance with the configuration provided and the characteristics of the data.

Short read data

Each line corresponds to a sample. There are four required fields: Sample name, strandness, RNA-seq paired data, merge. Followed by three optional fields: score, is_reference, exclude_redundant. Previous fields to an optional field must be present in the line. Files within a pair are separated by semi-colon and where there are multiple pairs in a sample, these are separated by spaces.

```
Ara0,fr-firststrand,data/Ara1.1.fastq.gz;data/Ara1.2.fastq.gz,true,20
Ara1,fr-firststrand,data/Ara1.1.fastq.gz;data/Ara1.2.fastq.gz data/Ara2.1.fastq.gz;data/
↪Ara2.2.fastq.gz,true,20
Ara2,fr-firststrand,data/Ara3.1.fastq.gz;data/Ara3.2.fastq.gz data/Ara5.1.fastq.gz;data/
↪Ara5.2.fastq.gz data/Ara6.1.fastq.gz;data/Ara6.2.fastq.gz,false
```

Sample RNA-seq data can be merged in different places, the options for controlling when the merging happens are as follows: All transcripts assembled from paired reads within a sample are combined after assembling, paired read alignments can be merged before assembly using the 'merge' parameter in the CSV file.

Junctions

Junctions from RNA-seq data can be determined in several ways. By default junctions are collected for all the RNA-seq fastq pair as defined in the 'RNA-seq paired data' section of the CSV file for each sample. Alternatively, samples can be combined where appropriate using the 'ei_annotation.wf_align.group_to_samples' parameter in the input.json file. This parameter will define arbitrary groupings of the samples present in the short read CSV, with the following format:

```
"ei_annotation.wf_align.group_to_samples": {
  "group1": ["Sample1", "Sample2"],
  "group2": ["Sample3", "Sample4"]
}
```

These groups will be validated against the samples in the CSV files, group names should be unique, samples can only belong to a single group and all samples should be part of a group.

Long read data

Each line corresponds to a sample. There are four required fields: Sample name, strandness, RNA-seq long read data, merge. Followed by three optional fields: score, is_reference and exclude_redundant. Previous fields to an optional field must be present in the line. Where multiple read files correspond to a single sample (this implies they result in a single set of transcripts), the third column will contain all the files separated by spaces.

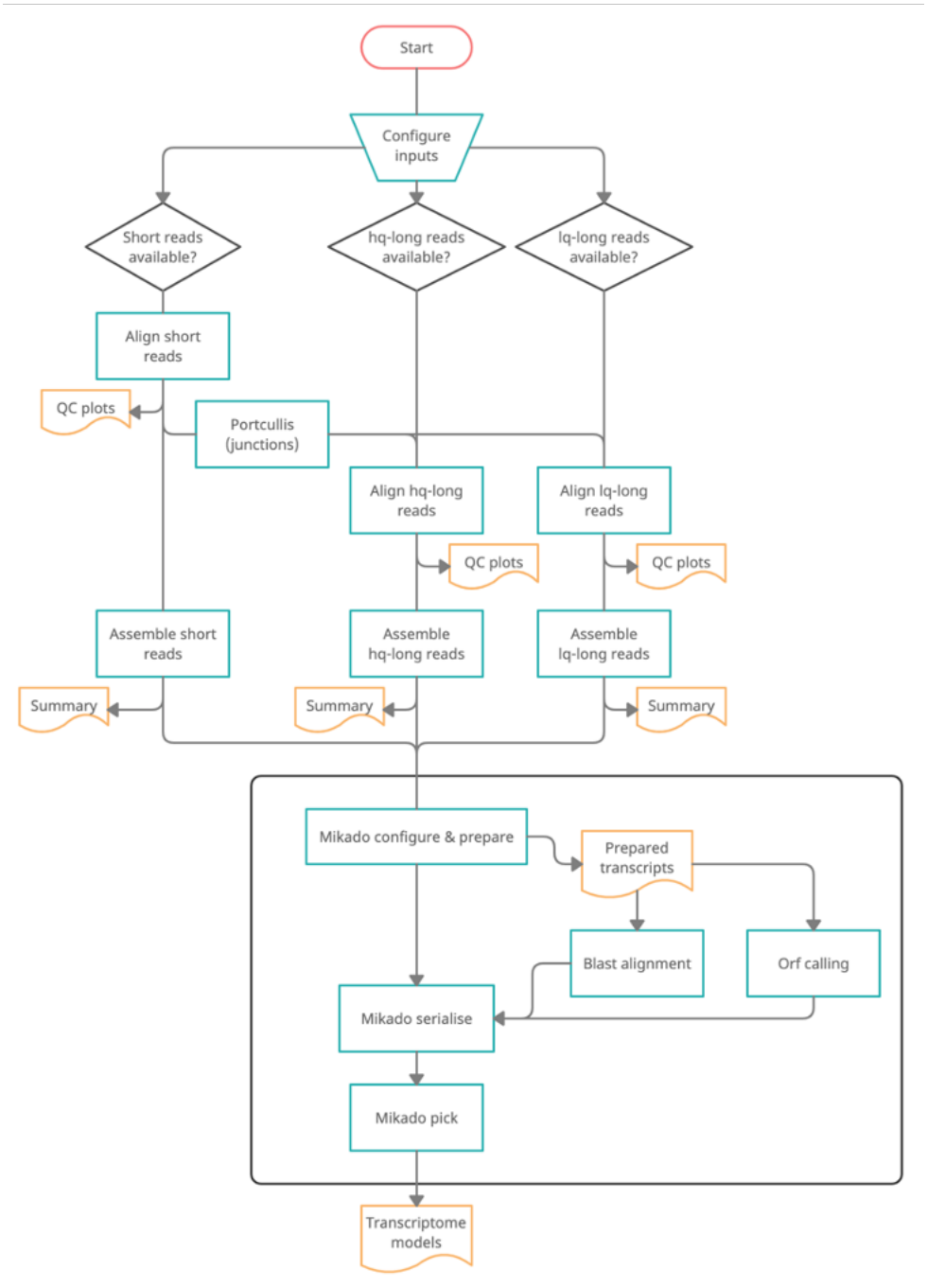
```
A01_1,fr-firststrand,data/A1_1.fastq.gz,low
A01_2,fr-firststrand,data/A1_2.fastq.gz,low
B01,fr-firststrand,data/B1.fastq.gz,low,10,true,true
C01,fr-firststrand,data/C1.fastq.gz,low
ALL,fr-firststrand,data/D1_1.fastq.gz data/D1_2.fastq.gz data/D1_3.fastq.gz data/D1_4.
↪fastq.gz,low
```

(continues on next page)

(continued from previous page)

```
CCS,fr-firststrand,data/CCS.fastq.gz,high
polished,fr-firststrand,data/polished.fastq.gz,high
```

Warning: The 'reference' sample name is reserved for internal use. If this name is being used in any of the sample input CSV files, you will be notified with an error message.



1.1.2 Configurable computational resources available

```
"ei_annotation.wf_align.long_read_alignment_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.long_read_assembly_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.long_read_indexing_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.long_read_twopass_merge_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.long_read_twopass_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.portcullis_resources": " {
```

(continues on next page)

(continued from previous page)

```
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.sanitise_reference_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.short_read_alignment_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.short_read_alignment_sort_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.short_read_indexing_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_annotation.wf_align.short_read_merge_resources": " {
                cpu_cores -> Int?
               max_retries -> Int?
```

(continues on next page)

(continued from previous page)

```
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_align.short_read_scallop_assembly_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_align.short_read_stats_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_align.short_read_stringtie_assembly_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.homology_alignment_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.homology_index_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
```

(continues on next page)

(continued from previous page)

```
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.mikado_all_pick_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.mikado_all_prepare_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.mikado_all_serialise_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.mikado_long_lq_pick_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
                mem_gb -> Float?
                preemptible_tries -> Int?
                }? (optional)",
"ei_annotation.wf_main_mikado.mikado_long_lq_prepare_resources": " {
                 cpu_cores -> Int?
                max_retries -> Int?
                boot_disk_gb -> Int?
                queue -> String?
                disk_gb -> Int?
                constraints -> String?
```

(continues on next page)

(continued from previous page)

```
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_annotation.wf_main_mikado.mikado_long_lq_serialise_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_annotation.wf_main_mikado.mikado_long_pick_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_annotation.wf_main_mikado.mikado_long_prepare_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_annotation.wf_main_mikado.mikado_long_serialise_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_annotation.wf_main_mikado.orf_calling_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
```

(continues on next page)

(continued from previous page)

```
              }? (optional)",
"ei_annotation.wf_main_mikado.protein_alignment_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_annotation.wf_main_mikado.protein_index_resources": " {
               cpu_cores -> Int?
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)"
```

1.2 Homology Workflow

```
Welcome to REAT
version - 0.6.1

Command-line call:
/home/docs/checkouts/readthedocs.org/user_builds/reat/envs/stable/bin/reat homology --
↪help


usage: reat homology [-h] --genome GENOME [-p OUTPUT_PREFIX]
                     --alignment_species ALIGNMENT_SPECIES
                     [--annotations_csv ANNOTATIONS_CSV]
                     [--protein_sequences [PROTEIN_SEQUENCES [PROTEIN_SEQUENCES ...]]]
                     [--annotation_filters {all,none,exon_len,intron_len,internal_stop,
↪aa_len,splicing} [{all,none,exon_len,intron_len,internal_stop,aa_len,splicing} ...]]
                     --mikado_config MIKADO_CONFIG --mikado_scoring
                     MIKADO_SCORING [--junctions JUNCTIONS] [--utrs UTRS]
                     [--pick_extra_config PICK_EXTRA_CONFIG]
                     [--min_cdna_length MIN_CDNA_LENGTH]
                     [--max_intron_length MAX_INTRON_LENGTH]
                     [--filter_min_cds FILTER_MIN_CDS]
                     [--filter_max_intron FILTER_MAX_INTRON]
                     [--filter_min_exon FILTER_MIN_EXON]
                     [--alignment_min_exon_len ALIGNMENT_MIN_EXON_LEN]
                     [--alignment_filters {all,none,exon_len,intron_len,internal_stop,aa_
↪len,splicing} [{all,none,exon_len,intron_len,internal_stop,aa_len,splicing} ...]]
                     [--alignment_min_identity ALIGNMENT_MIN_IDENTITY]
```

(continues on next page)

(continued from previous page)

```
                     [--alignment_min_coverage ALIGNMENT_MIN_COVERAGE]
                     [--alignment_max_per_query ALIGNMENT_MAX_PER_QUERY]
                     [--alignment_recursion_level ALIGNMENT_RECURSION_LEVEL]
                     [--alignment_show_intron_length]
                     [--exon_f1_filter EXON_F1_FILTER]
                     [--junction_f1_filter JUNCTION_F1_FILTER]

optional arguments:
  -h, --help            show this help message and exit
  --genome GENOME       Fasta file of the genome to annotate (default: None)
  -p OUTPUT_PREFIX, --output_prefix OUTPUT_PREFIX
                        Prefix for the final output files (default: xspecies)
  --alignment_species ALIGNMENT_SPECIES
                        Species specific parameters, select a value from the first or␣
↪second column of https://raw.githubusercontent.com/ogotoh/spaln/master/table/gnm2tab␣
↪(default: None)
  --annotations_csv ANNOTATIONS_CSV
                        CSV file with reference annotations to extract proteins/cdnas␣
↪for spliced alignments. The CSV fields are: genome_fasta,annotation_gff
                        Example:
                        Athaliana.fa,Athaliana.gff (default: None)
  --protein_sequences [PROTEIN_SEQUENCES [PROTEIN_SEQUENCES ...]]
                        List of files containing protein sequences to use as evidence␣
↪(default: None)
  --annotation_filters {all,none,exon_len,intron_len,internal_stop,aa_len,splicing} [
↪{all,none,exon_len,intron_len,internal_stop,aa_len,splicing} ...]
                        Filter annotation coding genes by the filter types specified␣
↪(default: ['none'])
  --mikado_config MIKADO_CONFIG
                        Base configuration for Mikado consolidation stage. (default:␣
↪None)
  --mikado_scoring MIKADO_SCORING
                        Scoring file for Mikado pick at consolidation stage. (default:␣
↪None)
  --junctions JUNCTIONS
                        Validated junctions BED file for use in Mikado consolidation␣
↪stage. (default: None)
  --utrs UTRS           Gene models that may provide UTR extensions to the homology␣
↪based models at the mikado stage (default: None)
  --pick_extra_config PICK_EXTRA_CONFIG
                        Extra configuration for Mikado pick stage (default: None)
  --min_cdna_length MIN_CDNA_LENGTH
                        Minimum cdna length for models to consider in Mikado␣
↪consolidation stage (default: 100)
  --max_intron_length MAX_INTRON_LENGTH
                        Maximum intron length for models to consider in Mikado␣
↪consolidation stage (default: 1000000)
  --filter_min_cds FILTER_MIN_CDS
                        If 'aa_len' filter is enabled for annotation coding features,␣
↪any CDS smaller thanthis parameter will be filtered out (default: 20)
  --filter_max_intron FILTER_MAX_INTRON
                        If 'intron_len' filter is enabled, any features with introns␣
↪longer than this parameter will be filtered out (default: 200000)
```

(continues on next page)

(continued from previous page)

```
  --filter_min_exon FILTER_MIN_EXON
                        If 'exon_len' filter is enabled, any features with exons shorter␣
↪than this parameter will be filtered out (default: 20)
  --alignment_min_exon_len ALIGNMENT_MIN_EXON_LEN
                        Minimum exon length, alignment parameter (default: 20)
  --alignment_filters {all,none,exon_len,intron_len,internal_stop,aa_len,splicing} [{all,
↪none,exon_len,intron_len,internal_stop,aa_len,splicing} ...]
                        Filter alignment results by the filter types specified (default:␣
↪['none'])
  --alignment_min_identity ALIGNMENT_MIN_IDENTITY
                        Minimum identity filter for alignments (default: 50)
  --alignment_min_coverage ALIGNMENT_MIN_COVERAGE
                        Minimum coverage filter for alignments (default: 80)
  --alignment_max_per_query ALIGNMENT_MAX_PER_QUERY
                        Maximum number of alignments per input query protein (default: 4)
  --alignment_recursion_level ALIGNMENT_RECURSION_LEVEL
                        SPALN's Q value, indicating the level of recursion for the␣
↪Hirschberg algorithm (default: 6)
  --alignment_show_intron_length
                        Add an attribute to the alignment gff with the maximum intron␣
↪len for each mRNA (default: False)
  --exon_f1_filter EXON_F1_FILTER
                        Filter alignments scored against its original structure with a␣
↪CDS exon f1 lower than this value (default: None)
  --junction_f1_filter JUNCTION_F1_FILTER
                        Filter alignments scored against its original structure with a␣
↪CDS junction f1 lower than this value (default: None)
```

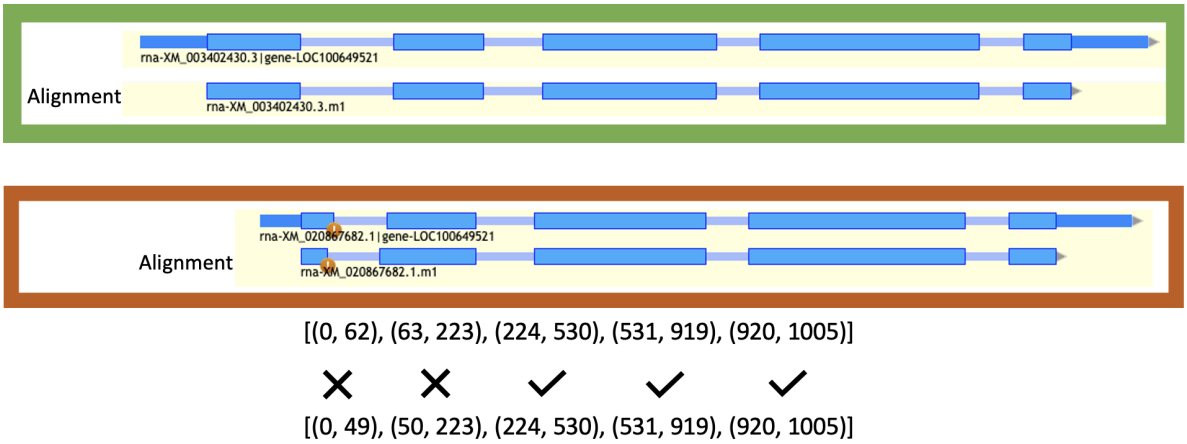
When there is protein evidence available from related species, the homology workflow can be used to generate gene models based on this evidence. This is achieved by aligning the proteins provided through a set of related species annotations and evaluating these alignments to generate a score.

After the proteins from related species are aligned to the reference, these alignments are filtered using a set of selectable criteria such as: exon length, intron length, protein length, canonical splicing and presence of internal stop codons. The resulting sequences are then evaluated.

Protein alignments are evaluated in two ways, coherence of the alignment structure with respect to the original model's structure and consensus structure from the multiple species. These scores are then used by Mikado to group and filter models, generating a set of predicted models.

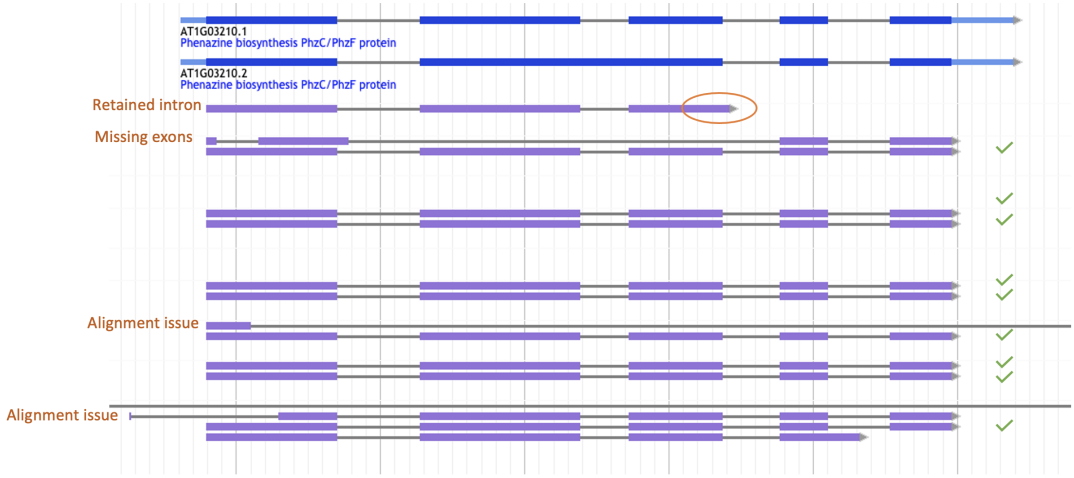
1.2.1 Comparing original structure vs alignment result

In the following example we observe two cases of the comparison between alignments and the original structure, the example on the top of the image shows the original gene structure matching that of the protein alignment structure. On the other hand, on the bottom of the image the original structure does not match the aligned protein, specifically the first and second exons have different lengths.



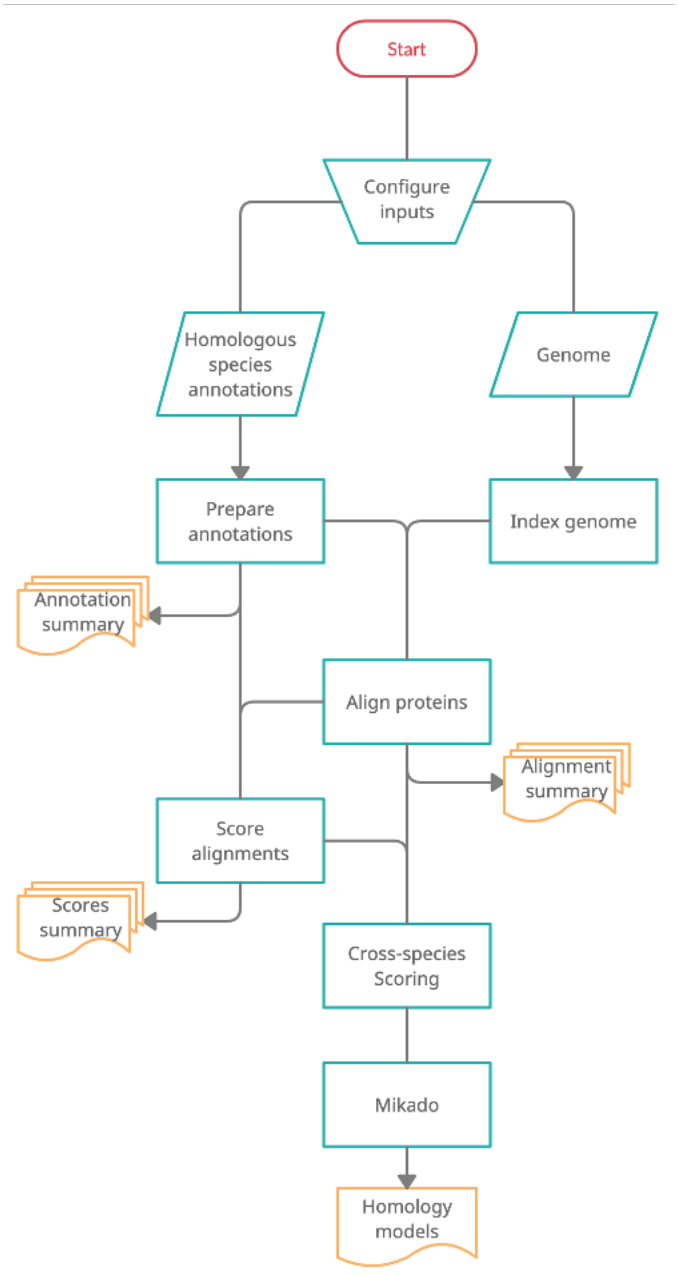
1.2.2 Cross species scoring

When comparing alignments from multiple species we evaluate the consensus structure and provide a score corresponding to the most commonly observed structure. In the following example, we observe a clear majority of protein alignments supporting a structure with five exons, this leads to a **xspecies** score of 70% for the models marked correct.



1.2.3 Configurable computational resources available

```
"ei_homology.CombineXspecies.runtime_attr_override": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_homology.aln_attr": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_homology.index_attr": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_homology.mikado_attr": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_homology.score_attr": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)"
```



1.3 Prediction Workflow

The intention of the prediction workflow is to use a variety of transcript evidence, from short reads and long reads based gene assemblies, protein alignments, homology alignments and other evidence such as expression, introns and repeats to generate gene predictions ab initio and evidence based gene predictions.

```
Welcome to REAT
version - 0.6.1

Command-line call:
/home/docs/checkouts/readthedocs.org/user_builds/reat/envs/stable/bin/reat prediction --
↪help
```

(continues on next page)

(continued from previous page)

```
usage: reat prediction [-h] --genome GENOME --augustus_config_path
                       AUGUSTUS_CONFIG_PATH
                       [--extrinsic_config [EXTRINSIC_CONFIG [EXTRINSIC_CONFIG ...]]]
                       --species SPECIES [--chunk_size CHUNK_SIZE]
                       [--overlap_size OVERLAP_SIZE]
                       [--transcriptome_models [TRANSCRIPTOME_MODELS [TRANSCRIPTOME_
↪MODELS ...]]]
                       [--homology_models [HOMOLOGY_MODELS [HOMOLOGY_MODELS ...]]]
                       [--introns INTRONS]
                       [--firststrand_expression FIRSTSTRAND_EXPRESSION]
                       [--secondstrand_expression SECONDSTRAND_EXPRESSION]
                       [--unstranded_expression UNSTRANDED_EXPRESSION]
                       [--repeats REPEATS] --homology_proteins
                       HOMOLOGY_PROTEINS [--optimise_augustus] [--kfold KFOLD]
                       [--force_train]
                       [--augustus_runs [AUGUSTUS_RUNS [AUGUSTUS_RUNS ...]]]
                       --EVM_weights EVM_WEIGHTS
                       [--hq_protein_alignments [HQ_PROTEIN_ALIGNMENTS [HQ_PROTEIN_
↪ALIGNMENTS ...]]]
                       [--lq_protein_alignments [LQ_PROTEIN_ALIGNMENTS [LQ_PROTEIN_
↪ALIGNMENTS ...]]]
                       [--hq_assembly [HQ_ASSEMBLY [HQ_ASSEMBLY ...]]]
                       [--lq_assembly [LQ_ASSEMBLY [LQ_ASSEMBLY ...]]]
                       [--mikado_utr_files [{gold,silver,bronze,all,hq_assembly,lq_
↪assembly} [{gold,silver,bronze,all,hq_assembly,lq_assembly} ...]]]
                       [--do_glimmer [DO_GLIMMER]] [--do_snap [DO_SNAP]]
                       [--do_codingquarry [DO_CODINGQUARRY]] [--no_augustus]
                       [--filter_top_n FILTER_TOP_N]
                       [--filter_max_identity FILTER_MAX_IDENTITY]
                       [--filter_max_coverage FILTER_MAX_COVERAGE]
                       [--codingquarry_extra_params CODINGQUARRY_EXTRA_PARAMS]
                       [--glimmer_extra_params GLIMMER_EXTRA_PARAMS]
                       [--snap_extra_params SNAP_EXTRA_PARAMS]
                       [--augustus_extra_params AUGUSTUS_EXTRA_PARAMS]
                       [--evm_extra_params EVM_EXTRA_PARAMS]
                       [--min_train_models MIN_TRAIN_MODELS]
                       [--max_train_models MAX_TRAIN_MODELS]
                       [--max_test_models MAX_TEST_MODELS]
                       [--target_mono_exonic_percentage TARGET_MONO_EXONIC_PERCENTAGE]
                       [--force_train_few_models]
                       [--evalue_filter EVALUE_FILTER]
                       [--min_pct_cds_fraction MIN_PCT_CDS_FRACTION]
                       [--max_tp_utr_complete MAX_TP_UTR_COMPLETE]
                       [--max_tp_utr MAX_TP_UTR] [--min_tp_utr MIN_TP_UTR]
                       [--max_fp_utr_complete MAX_FP_UTR_COMPLETE]
                       [--max_fp_utr MAX_FP_UTR] [--min_fp_utr MIN_FP_UTR]
                       [--query_start_hard_filter_distance QUERY_START_HARD_FILTER_
↪DISTANCE]
                       [--query_start_score QUERY_START_SCORE]
                       [--query_start_scoring_distance QUERY_START_SCORING_DISTANCE]
```

(continues on next page)

(continued from previous page)

```
                        [--query_end_hard_filter_distance QUERY_END_HARD_FILTER_DISTANCE]
                        [--query_end_score QUERY_END_SCORE]
                        [--query_end_scoring_distance QUERY_END_SCORING_DISTANCE]
                        [--target_start_hard_filter_distance TARGET_START_HARD_FILTER_
↪DISTANCE]
                        [--target_start_score TARGET_START_SCORE]
                        [--target_start_scoring_distance TARGET_START_SCORING_DISTANCE]
                        [--target_end_hard_filter_distance TARGET_END_HARD_FILTER_
↪DISTANCE]
                        [--target_end_score TARGET_END_SCORE]
                        [--target_end_scoring_distance TARGET_END_SCORING_DISTANCE]
                        [--min_query_coverage_hard_filter MIN_QUERY_COVERAGE_HARD_FILTER]
                        [--min_query_coverage_score MIN_QUERY_COVERAGE_SCORE]
                        [--min_query_coverage_scoring_percentage MIN_QUERY_COVERAGE_
↪SCORING_PERCENTAGE]
                        [--min_target_coverage_hard_filter MIN_TARGET_COVERAGE_HARD_
↪FILTER]
                        [--min_target_coverage_score MIN_TARGET_COVERAGE_SCORE]
                        [--min_target_coverage_scoring_percentage MIN_TARGET_COVERAGE_
↪SCORING_PERCENTAGE]
                        [--max_single_gap_hard_filter MAX_SINGLE_GAP_HARD_FILTER]
                        [--max_single_gap_score MAX_SINGLE_GAP_SCORE]
                        [--max_single_gap_scoring_length MAX_SINGLE_GAP_SCORING_LENGTH]

optional arguments:
  -h, --help            show this help message and exit
  --genome GENOME       Genome fasta file (default: None)
  --augustus_config_path AUGUSTUS_CONFIG_PATH
                        Template path for augustus config, this path will not be␣
↪modified as a copy will be created internally for the workflow's use (default: None)
  --extrinsic_config [EXTRINSIC_CONFIG [EXTRINSIC_CONFIG ...]]
                        Augustus extrinsic configuration file, defines the boni/mali for␣
↪each type of feature-evidence combination (default: None)
  --species SPECIES     Name of the species to train models for, if it does not exist in␣
↪the augustus config path it will be created. (default: None)
  --chunk_size CHUNK_SIZE
                        Maximum length of sequence to be processed by Augustus or EVM␣
↪(default: 3000000)
  --overlap_size OVERLAP_SIZE
                        Overlap length for sequences longer than chunk_size for EVM and␣
↪Augustus (default: 100000)
  --transcriptome_models [TRANSCRIPTOME_MODELS [TRANSCRIPTOME_MODELS ...]]
                        Models derived from transcriptomic data (default: None)
  --homology_models [HOMOLOGY_MODELS [HOMOLOGY_MODELS ...]]
                        Models derived from protein alignments (default: None)
  --introns INTRONS     Introns to be used as hints for Augustus (default: None)
  --firststrand_expression FIRSTSTRAND_EXPRESSION
                        Sorted by position first-strand RNASeq alignments used for␣
↪coverage hints (default: None)
  --secondstrand_expression SECONDSTRAND_EXPRESSION
                        Sorted by position second-strand RNAseq alignments used for␣
↪coverage hints (default: None)
```

(continues on next page)

(continued from previous page)

```
  --unstranded_expression UNSTRANDED_EXPRESSION
                        Sorted by position unstranded RNAseq alignments used for␣
↪coverage hints (default: None)
  --repeats REPEATS     Repeat annotation GFF file. (default: None)
  --homology_proteins HOMOLOGY_PROTEINS
                        Protein DB of sequences used for determining whether the␣
↪evidence provided is full-length or not (default: None)
  --optimise_augustus   Enable augustus metaparameter optimisation (default: False)
  --kfold KFOLD         Number of batches for augustus optimisation (default: 8)
  --force_train         Re-train augustus even if the species is found in the 'augustus_
↪config_path' (default: False)
  --augustus_runs [AUGUSTUS_RUNS [AUGUSTUS_RUNS ...]]
                        File composed of 13 lines with SOURCE PRIORITY pairs for each of␣
↪the types of evidence that can be used in
                        an Augustus run. These evidence types are: gold models, silver␣
↪models, bronze models, all models,
                        gold introns, silver introns, protein models, coverage hints,␣
↪repeat hints, high quality assemblies,
                        low quality assemblies, high quality proteins, and low quality␣
↪proteins. (default: None)
  --EVM_weights EVM_WEIGHTS
                        Evidence modeler requires a weighting to be provided for each␣
↪source of evidence, this file is the means to do so. (default: None)
  --hq_protein_alignments [HQ_PROTEIN_ALIGNMENTS [HQ_PROTEIN_ALIGNMENTS ...]]
                        High confidence protein alignments to be used as hints for␣
↪Augustus runs (default: None)
  --lq_protein_alignments [LQ_PROTEIN_ALIGNMENTS [LQ_PROTEIN_ALIGNMENTS ...]]
                        Low confidence protein alignments to be used as hints for␣
↪Augustus runs (default: None)
  --hq_assembly [HQ_ASSEMBLY [HQ_ASSEMBLY ...]]
                        High confidence assemblies (for example from HiFi source) to be␣
↪used as hints for Augustus runs (default: None)
  --lq_assembly [LQ_ASSEMBLY [LQ_ASSEMBLY ...]]
                        Low confidence assemblies (short reads or low quality long␣
↪reads) to be used as hints for Augustus runs (default: None)
  --mikado_utr_files [{gold,silver,bronze,all,hq_assembly,lq_assembly} [{gold,silver,
↪bronze,all,hq_assembly,lq_assembly} ...]]
                        Choose any combination of space separated values from: gold␣
↪silver bronze all hq_assembly lq_assembly (default: ['gold', 'silver'])
  --do_glimmer [DO_GLIMMER]
                        Enables GlimmerHmm predictions, optionally accepts a training␣
↪directory (default: None)
  --do_snap [DO_SNAP]   Enables SNAP predictions, optionally accepts a training␣
↪directory (default: None)
  --do_codingquarry [DO_CODINGQUARRY]
                        Enables CodingQuarry predictions, optionally accepts a training␣
↪directory (default: None)
  --no_augustus
  --filter_top_n FILTER_TOP_N
                        Only output the top N transcripts that pass the self blast␣
↪filter (0 outputs all) (default: 0)
  --filter_max_identity FILTER_MAX_IDENTITY
```

(continues on next page)

(continued from previous page)

```
                        Maximum identity between models for redundancy classification␣
↪(default: 80)
  --filter_max_coverage FILTER_MAX_COVERAGE
                        Maximum coverage between models for redundancy classification␣
↪(default: 80)
  --codingquarry_extra_params CODINGQUARRY_EXTRA_PARAMS
                        Extra parameters for CodingQuarry predictions (default: None)
  --glimmer_extra_params GLIMMER_EXTRA_PARAMS
                        Extra parameters for glimmer predictions (default: None)
  --snap_extra_params SNAP_EXTRA_PARAMS
                        Extra parameters for snap predictions (default: None)
  --augustus_extra_params AUGUSTUS_EXTRA_PARAMS
                        Extra parameters for all Augustus predictions (default: None)
  --evm_extra_params EVM_EXTRA_PARAMS
                        Extra parameters for EVM gene predictions consolidation␣
↪(default: None)
  --min_train_models MIN_TRAIN_MODELS
                        Minimum number of training models (default: 400)
  --max_train_models MAX_TRAIN_MODELS
                        Maximum number of training models (default: 1000)
  --max_test_models MAX_TEST_MODELS
                        Maximum number of test models (default: 200)
  --target_mono_exonic_percentage TARGET_MONO_EXONIC_PERCENTAGE
                        Target percentage of mono-exonic models in the training set␣
↪(default: 20)
  --force_train_few_models
                        Train Augustus regardless monoexonic model ratio and number of␣
↪models (default: False)
  --evalue_filter EVALUE_FILTER
                        Hits with a value higher than this will be filtered out of the␣
↪initial set (default: 1e-06)
  --min_pct_cds_fraction MIN_PCT_CDS_FRACTION
                        Transcript requirement for minimum faction of transcript covered␣
↪by CDS (default: 0.5)
  --max_tp_utr_complete MAX_TP_UTR_COMPLETE
                        Transcript requirement for maximum complete 3' UTRs (default: 1)
  --max_tp_utr MAX_TP_UTR
                        Transcript requirement for maximum number of 3' UTRs (default: 2)
  --min_tp_utr MIN_TP_UTR
                        Transcript requirement for minimum number of 3' UTRs (default: 1)
  --max_fp_utr_complete MAX_FP_UTR_COMPLETE
                        Transcript requirement for maximum number of complete 5' UTRs␣
↪(default: 2)
  --max_fp_utr MAX_FP_UTR
                        Transcript requirement for maximum number of 5' UTRs (default: 3)
  --min_fp_utr MIN_FP_UTR
                        Transcript requirement for minimum number of 5' UTRs (default: 1)
  --query_start_hard_filter_distance QUERY_START_HARD_FILTER_DISTANCE
                        If query hit starts after this value, the transcript cannot␣
↪belong to the Gold category (default: 10)
  --query_start_score QUERY_START_SCORE
                        Maximum score for query start distance (default: 5)
```

(continues on next page)

(continued from previous page)

```
  --query_start_scoring_distance QUERY_START_SCORING_DISTANCE
                        Hits with query start distance lower than this parameter start␣
↪receiving scoring points (default: 30)
  --query_end_hard_filter_distance QUERY_END_HARD_FILTER_DISTANCE
                        If query hit ends after this value, the transcript cannot belong␣
↪to the Gold category (default: 10)
  --query_end_score QUERY_END_SCORE
                        Maximum score for query end distance (default: 5)
  --query_end_scoring_distance QUERY_END_SCORING_DISTANCE
                        Hits with query end distance lower than this parameter start␣
↪receiving scoring points (default: 30)
  --target_start_hard_filter_distance TARGET_START_HARD_FILTER_DISTANCE
                        If target hit starts after this value, the transcript cannot␣
↪belong to the Gold category (default: 10)
  --target_start_score TARGET_START_SCORE
                        Maximum score for target start distance (default: 5)
  --target_start_scoring_distance TARGET_START_SCORING_DISTANCE
                        Hits with target start distance lower than this parameter start␣
↪receiving scoring points (default: 30)
  --target_end_hard_filter_distance TARGET_END_HARD_FILTER_DISTANCE
                        If target hit ends after this value, the transcript cannot␣
↪belong to the Gold category (default: 10)
  --target_end_score TARGET_END_SCORE
                        Maximum score for target end distance (default: 5)
  --target_end_scoring_distance TARGET_END_SCORING_DISTANCE
                        Hits with target end distance lower than this parameter start␣
↪receiving scoring points (default: 30)
  --min_query_coverage_hard_filter MIN_QUERY_COVERAGE_HARD_FILTER
                        Minimum percentage of query covered to classify a hit as Gold␣
↪(default: 90)
  --min_query_coverage_score MIN_QUERY_COVERAGE_SCORE
                        Maximum score for query percentage coverage (default: 5)
  --min_query_coverage_scoring_percentage MIN_QUERY_COVERAGE_SCORING_PERCENTAGE
                        Queries covered over this percentage value start receiving␣
↪scoring points (default: 30)
  --min_target_coverage_hard_filter MIN_TARGET_COVERAGE_HARD_FILTER
                        Minimum percentage of target covered to classify a hit as Gold␣
↪(default: 90)
  --min_target_coverage_score MIN_TARGET_COVERAGE_SCORE
                        Maximum score for target percentage coverage (default: 5)
  --min_target_coverage_scoring_percentage MIN_TARGET_COVERAGE_SCORING_PERCENTAGE
                        Targets covered over this percentage value start receiving␣
↪scoring points (default: 30)
  --max_single_gap_hard_filter MAX_SINGLE_GAP_HARD_FILTER
                        Any hits containing gaps larger than this parameter cannot be␣
↪classified as Gold (default: 20)
  --max_single_gap_score MAX_SINGLE_GAP_SCORE
                        Maximum score for hits with single gaps smaller than --max_
↪single_gap_scoring_length (default: 5)
  --max_single_gap_scoring_length MAX_SINGLE_GAP_SCORING_LENGTH
                        Hits containing gaps smaller than this parameter start receiving␣
↪scoring points (default: 30)
```

(continues on next page)

(continued from previous page)

The prediction module takes as input a genome file along with a set of evidences for annotations over the genome (these should have gene->mrna->{exon,CDS} structure, where CDS is required for protein inputs), these can come from homology proteins or transcript alignments, rna-seq gene models, repeat annotations, rna-seq alignments which can provide evidence to the presence/absence of exons. Also, the user should provide a set of proteins to validate against, these proteins are used to score input models, categorize them into Gold, Silver or Bronze and select the best models for training of the ab initio gene predictors.

Multiple sets of input models from homology proteins or transcriptomic sources are aligned to a protein database of the user's choice and the results of these alignments are used to classify and score each input model into Bronze, Silver and Gold. Models from the Gold and Silver category are defined by:

- Having complete but not excessively long UTR's.
- Being fully covered by multiple proteins from the database.
- Having a long enough CDS, where the length is user defined.

For scoring the models, a score is calculated for the following properties: the distance between the start and end of the model and the target protein are compared to the start and end of the alignment; the coverage of the model and the target protein; and the length of the longest gap in the alignment. The score is defined by three parameters that are user controlled, a 'hard filter' after which the criteria is considered failed, a 'soft filter' from where alignments receive a score relative to the difference between the 'best' possible value and the current level, and finally the maximum possible score.

Once models have been scored, models with more than a coverage and identity user defined threshold (80% by default) are filtered. From the similarity filtered models, a user defined number of models at a user defined ratio between mono-exonic and multi-exonic are randomly selected to train ab initio predictors. These models are selected from the classified models ordered by 'category' (Gold, Silver, Bronze, others in this order) and score (highest to lowest).

Each of the ab initio predictors the user selected is then trained and used to generate predictions. In the case of Augustus, there is an initial ab initio prediction made with limited evidence, but further rounds of prediction with different weights for each evidence type can then be configured using a file containing a SOURCE and a SCORE value for each criteria (*see*). These parameters depend on the extrinsic information configuration file used by Augustus, for more information about REAT's default *see the following section*. All these predictions are then combined using Evidence Modeler with configurable weights for each type of prediction and evidence (*see*). Finally, the EVM output is processed through Mikado using the Gold and Silver category models (which contain UTRs) to add UTRs where evidence supports it.

Note: The EVM weights file should contain a line per prediction, in case of `--augustus_runs` there should be a line with a label and a weight for each Augustus run, the labels are fixed and have the form *AUGUSTUS_RUN#* where # corresponds to the position of the run file in the list of `--augustus_runs` provided through the command-line arguments.

An example weights file with three augustus runs would look like this: . OTHER_PREDICTION AUGUSTUS_RUN1 1 OTHER_PREDICTION AUGUSTUS_RUN2 1 OTHER_PREDICTION AUGUSTUS_RUN3 1 . . .

1.3.1 Configuring Augustus runs

When generating predictions using Augustus, we need to choose the weight parameters for each type of evidence, whilst at the same time possibly wanting to have multiple options of weight sets and priorities as to predict a comprehensive set of models that will maximise our chances of predicting correct structures. In REAT we can decide the number of Augustus predictions and the weights for each prediction using a configuration file per prediction. This file contains a pair of SOURCE and SCORE for each of the evidence types available, which are: gold models, silver models, bronze models, all models, gold introns, silver introns, protein models, coverage hints, repeat hints, high quality assemblies, low quality assemblies, high quality proteins, and low quality proteins. Each file provided to the `--augustus_runs` parameter will trigger a run of Augustus using the specific combination of weights and priorities defined for each evidence type, resulting in as many predictions as files provided.

Note: The output directory will contain a file of predictions corresponding to each `--augustus_runs` input files, these files are named *augustus_run#* where # corresponds to the position of the file in the command-line argument list of run files.

The default Augustus configuration file can be overridden to make available for the user different 'SOURCE's which can then be used for the `--augustus_runs` files, the following is an example of a 'run' file:

```
M 10
F 9
E 8
E 7
E 6
E 4
P 4
W 3
RM 1
E 2
E 2
E 2
E 2
```

Note:

The order of the features in this file is as follows:

- gold models
- silver models
- bronze models
- all models
- gold introns
- silver introns
- protein models
- coverage hints
- repeat hints
- high quality assemblies
- low quality assemblies

- high quality proteins
- low quality proteins

1.3.2 Extrinsic information configuration file

```
# extrinsic information configuration file for AUGUSTUS
# Mario Stanke (mstanke@gwdg.de)
# David Swarbreck (david.swarbreck@earlham.ac.uk)
# Gemy Kaithakottil (gemy.kaithakottil@earlham.ac.uk)


# source of extrinsic information:
# M manual anchor (required)
# P protein database hit
# E EST/cDNA database hit
# W wiggle track coverage info from RNA-Seq
# RM repeat hint

[SOURCES]
M RM F E P W

[SOURCE-PARAMETERS]
P individual_liability
E individual_liability
W individual_liability
F individual_liability
M individual_liability

[GENERAL]
      start        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
       stop        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
        tss        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
        tts        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
        ass        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
        dss        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
   exonpart        1   .992     M    1  1e+100  RM  1     1    F 1    1e4    E 1    1e2 ␣
↪  P 1    1    W 1    1.005
       exon        1   1        M    1  1e+100  RM  1     1    F 1    1e8    E 1    1e4 ␣
↪  P 1    1    W 1    1
 intronpart        1   1        M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1    W 1    1
     intron        1   0.01     M    1  1e+100  RM  1     1    F 1    1e8    E 1    1e4 ␣
↪  P 1    1e4  W 1    1
    CDSpart        1   1 0.985  M    1  1e+100  RM  1     1    F 1    1      E 1    1   ␣
↪  P 1    1e2  W 1    1
```

(continues on next page)

(continued from previous page)

```
        CDS         1   1         M    1  1e+100  RM  1     1    F 1    1       E 1    1   ␣
↪   P 1    1e4  W 1    1
    UTRpart         1   1 0.985  M    1  1e+100  RM  1     1    F 1    1       E 1    1   ␣
↪   P 1    1    W 1    1
        UTR         1   1         M    1  1e+100  RM  1     1    F 1    1       E 1    1   ␣
↪   P 1    1    W 1    1
     irpart         1   1         M    1  1e+100  RM  1     1    F 1    1       E 1    1   ␣
↪   P 1    1    W 1    1
nonexonpart         1   1         M    1  1e+100  RM  1     10   F 1    1       E 1    1   ␣
↪   P 1    1    W 1    1
  genicpart         1   1         M    1  1e+100  RM  1     1    F 1    1       E 1    1   ␣
↪   P 1    1    W 1    1

#
# Explanation:
#
# Please refer to the AUGUSTUS documentation for further details about this file
```

1.3.3 Evidence Modeler default weights file

```
ABINITIO_PREDICTION     GlimmerHMM      1
ABINITIO_PREDICTION     SNAP    1
ABINITIO_PREDICTION     CodingQuarry_v2.0       1
ABINITIO_PREDICTION AUGUSTUS_RUN_ABINITIO 1
PROTEIN hq_protein_alignment    4
PROTEIN lq_protein_alignment    1
TRANSCRIPT      hq_assembly     4
TRANSCRIPT      lq_asssembly    1
TRANSCRIPT      homology_models 10
TRANSCRIPT      transcriptome_models    10
OTHER_PREDICTION        AUGUSTUS_RUN1   10
OTHER_PREDICTION        AUGUSTUS_RUN2   10
OTHER_PREDICTION        AUGUSTUS_RUN3   10
```

1.3.4 Configurable computational resources available

```
      "ei_prediction.AlignProteins.resources": " {
               cpu_cores -> Int
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_prediction.Augustus.resources": " {
```

(continues on next page)

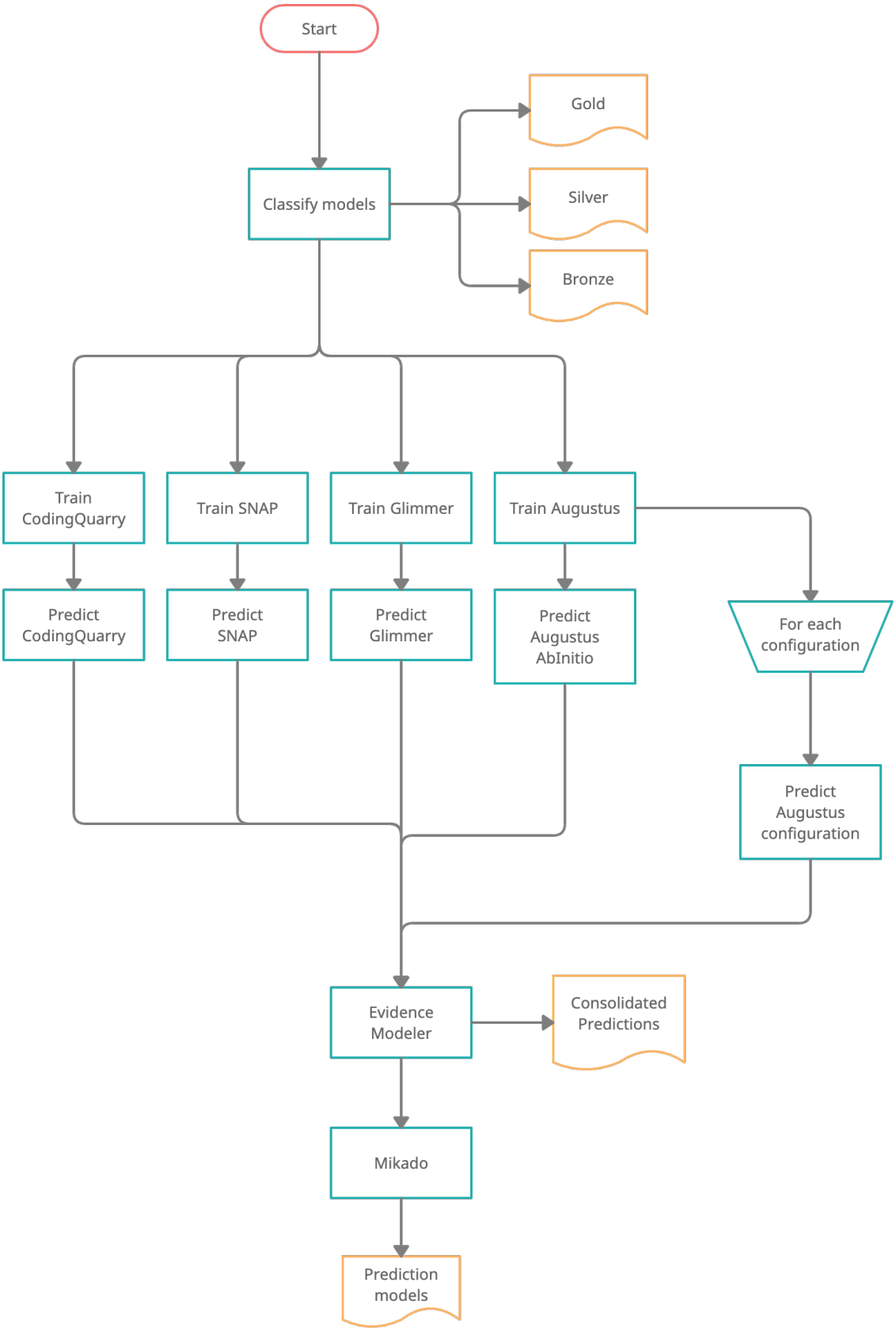
(continued from previous page)

```
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_prediction.AugustusAbinitio.resources": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_prediction.ExecuteEVMCommand.resources": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_prediction.IndexProteinsDatabase.resources": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_prediction.LengthChecker.resources": " {
                cpu_cores -> Int
               max_retries -> Int?
               boot_disk_gb -> Int?
               queue -> String?
               disk_gb -> Int?
               constraints -> String?
               mem_gb -> Float?
               preemptible_tries -> Int?
               }? (optional)",
"ei_prediction.Mikado.resources": " {
                cpu_cores -> Int
               max_retries -> Int?
```

(continues on next page)

(continued from previous page)

```
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_prediction.MikadoPick.resources": " {
               cpu_cores -> Int
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)",
"ei_prediction.SelfBlastFilter.resources": " {
               cpu_cores -> Int
              max_retries -> Int?
              boot_disk_gb -> Int?
              queue -> String?
              disk_gb -> Int?
              constraints -> String?
              mem_gb -> Float?
              preemptible_tries -> Int?
              }? (optional)"
```


Start
Classify models
Gold
Silver
Bronze
Train CodingQuarry
Train SNAP
Train Glimmer
Train Augustus
Predict CodingQuarry
Predict SNAP
Predict Glimmer
Predict Augustus AbInitio
For each configuration
Predict Augustus configuration
Evidence Modeler
Consolidated Predictions
Mikado
Prediction models

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search