REAT

Release 0.5.1

Luis Yanes

Dec 09, 2021

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
```

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"))

system.io {
  # For our shared cluster, you want to hit it with lots of requests and let SLURM␣
↪figure out priority, rather than waiting.
  number-of-requests = 1000000
  per = 1 seconds
  number-of-attempts = 5
}

system {
   file-hash-cache = true
   # Sometimes the defaults for input read limits were too small. These increase the max␣
↪file sizes.
   input-read-limits {
       tsv = 1073741823
       object = 1073741823
       string = 1073741823
       lines = 1073741823
       json = 1073741823
   }
}
```

```
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 - 0.5.1

Command-line call:
/home/docs/checkouts/readthedocs.org/user_builds/reat/envs/v0.5.1/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"]
}
```

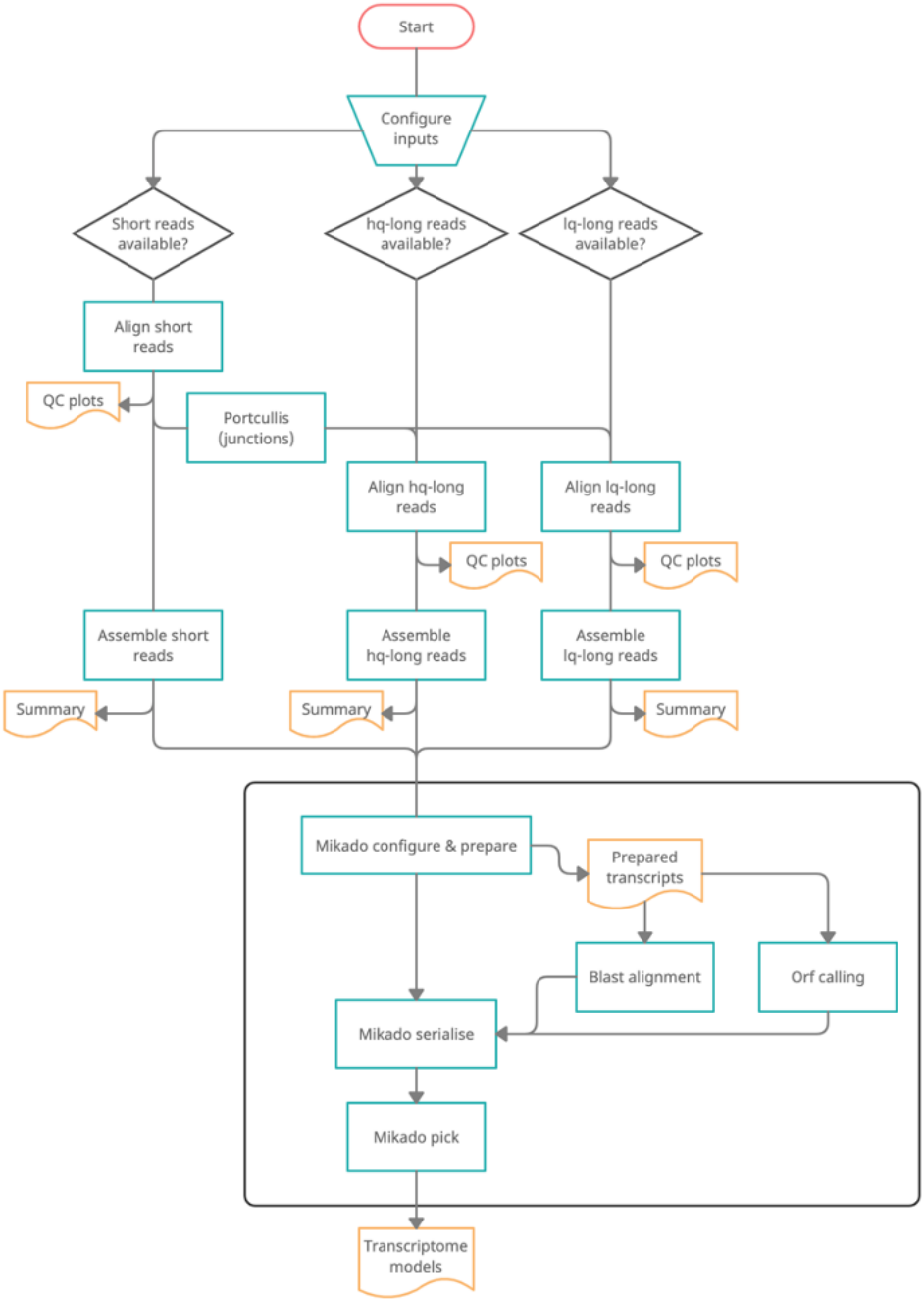
These groups will be validated against the samples in the CSV files, group names should be unique, samples can only belong to a single group and all samples should be part of a group.

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
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.



Configurable computational resources available:

```
{
    "ei_annotation.wf_align.long_read_alignment_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.long_read_assembly_resources":
```

(continues on next page)

(continued from previous page)

```
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.long_read_indexing_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.short_read_alignment_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.short_read_alignment_sort_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.short_read_merge_resources": {
        "cpu_cores": 4,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.short_read_scallop_assembly_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.short_read_stringtie_assembly_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_align.short_read_stats_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 8
    },
    "ei_annotation.wf_main_mikado.homology_alignment_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
```

(continues on next page)

(continued from previous page)

```
    "ei_annotation.wf_main_mikado.homology_index_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 8
    },
    "ei_annotation.wf_main_mikado.orf_calling_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 8
    },
    "ei_annotation.wf_main_mikado.protein_alignment_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    },
    "ei_annotation.wf_main_mikado.protein_index_resources":
    {
        "cpu_cores": 6,
        "max_retries": 1,
        "mem_gb": 16
    }
}
```

1.2 Homology Workflow

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.

```
Welcome to REAT
version - 0.5.1

Command-line call:
/home/docs/checkouts/readthedocs.org/user_builds/reat/envs/v0.5.1/bin/reat homology --
↪help


usage: reat homology [-h] --genome GENOME [-p OUTPUT_PREFIX]
                     --alignment_species ALIGNMENT_SPECIES
                     [--annotations_csv ANNOTATIONS_CSV]
```

(continues on next page)

(continued from previous page)

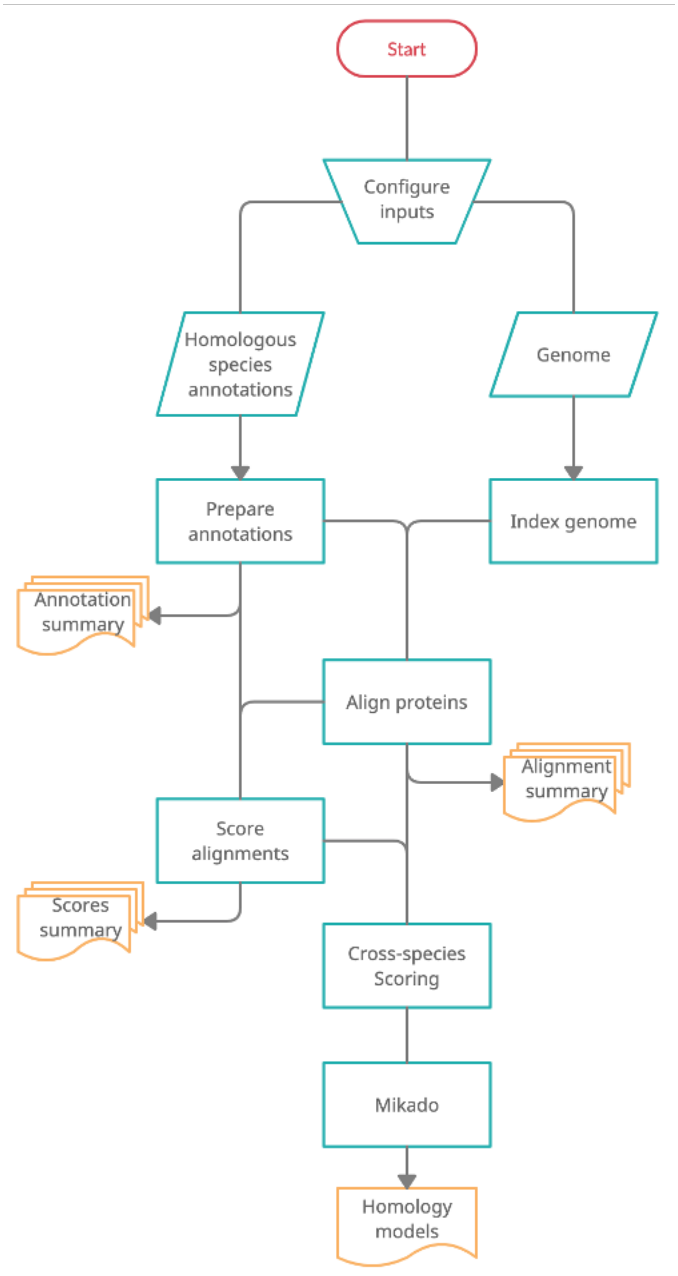
```
                     [--protein_sequences [PROTEIN_SEQUENCES [PROTEIN_SEQUENCES ...]]]
                     [--annotation_filters {all,none,exon_len,intron_len,internal_stop,
↪aa_len,splicing} [{all,none,exon_len,intron_len,internal_stop,aa_len,splicing} ...]]
                     --mikado_config 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]
                     [--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)
```

(continues on next page)

(continued from previous page)

```
  --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)
  --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)
```



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.

The prediction module takes as input a genome file along with a set of evidences for annotations over the genome, these can come from homology protein 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.

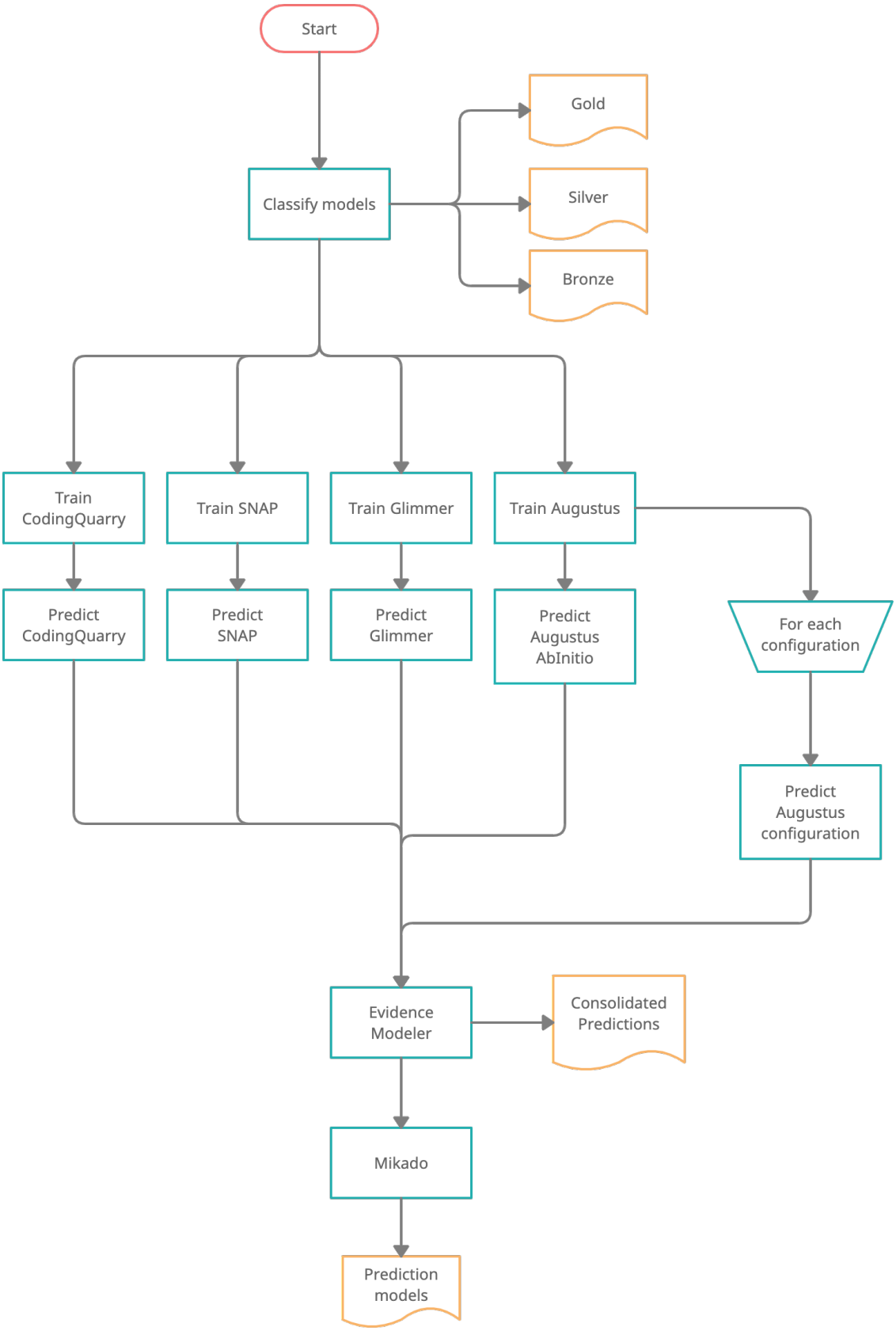
Input models from homology or transcriptomic sources are aligned to a protein database of the user's choice and the results of these alignments are used to classify and score each input model into Bronze, Silver and Gold. Models from the Gold and Silver category are defined by:

- 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, 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 (more details). All these predictions are then combined using Evidence Modeler with configurable weights for each type of prediction and evidence (more details). 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.


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