Annotation comparison Vetea Jacot

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 CDScompR Namespace Reference

This script is used to compute the distance between two structural annotations of a same genome, one reference and one alternative annotation.

Functions

- def write_results (all_results, alt_name, verbose=False)

  This function writes to a new 'results.csv' file the results of the annotation comparison retrieved from the identities dictionary returned by the annotation_match function.
- def annotation_comparison (ref_path, alt_path, verbose=False, create_strings=False, exon_mode=False)

  Main function of this program.
- def **usage** ()
- def **main** ()

3.1.1 Detailed Description

This script is used to compute the distance between two structural annotations of a same genome, one reference and one alternative annotation.

It expects as input the paths to the annotation files (in GFF format), displays the computed distances between all annotation pairs, and creates a results CSV file detailing the loci comparisons between the annotations

3.1.2 Function Documentation

3.1.2.1 annotation_comparison()

```
def CDScompR.annotation_comparison (
              ref_path,
              alt_path,
              verbose = False,
              create_strings = False,
              exon_mode = False )
```

Main function of this program.

Given a reference and alternative path, gets the corresponding GFF files and compares the two annotations to return their information about their loci's comparison

Parameters

ref_path	Path of the GFF file describing the reference annotation
alt_path	Path of the GFF file describing the aternative annotation
verbose	If True, triggers display of more information messages. Default is 'False'
create_strings	Boolean indicating wether to use the 'old' comparison function (old_compare_loci, 'True') or the new one (compare_loci, 'False')
exon_mode	Boolean indicating if the main comparison structures read from the file should be coding sequences (CDS, False) or exons (True). Default is 'False' (CDS comparison)

See also

compare_loci()

old_compare_loci()

Returns

Returns a list of lists of dictionaries describing the comparison of the structure identity between the loci of each annotation

3.1.2.2 write_results()

```
def CDScompR.write_results (
              all_results,
              alt_name,
              verbose = False )
```

This function writes to a new 'results.csv' file the results of the annotation comparison retrieved from the identities dictionary returned by the annotation_match function.

See also

annotation_match()

Parameters

results	A dictionary of list of list of dictionaries containing results of the annotation comparison, as returned by annotation_match
alt_name	String indicating the name of the original alternative annotation file
verbose	If True, triggers display of more information messages. Default is 'False'

See also

annotation_match()

Remarks

Results are written in CSV ('Comma-Separated Values') format

3.2 CDSmulticompR Namespace Reference

This script is used to compute the distance between multiple structural annotations of a same genome, one reference and one or more alternative annotations.

Functions

- def write_multi_results (multi_results, ref_path)

  Writes the results returned by the function multicomp into a results synthesis CSV file detailing the loci identity for each alternative.
- def multicomp (ref_path, alt_paths, verbose, create_strings, exon_mode)

  Compares all alternative annotations given (alt_paths) to the reference annotation (ref_path) by calling annotation_sort and appends the results dictionaries to a list.
- def **usage** ()
- def **main** ()

3.2.1 Detailed Description

This script is used to compute the distance between multiple structural annotations of a same genome, one reference and one or more alternative annotations.

It expects as input the paths to the annotation files (in GFF format), displays the computed distances between all annotation pairs, and creates results CSV files for each alternative and one global synthesis file

3.2.2 Function Documentation

3.2.2.1 multicomp()

```
def CDSmulticompR.multicomp (
                ref_path,
                alt_paths,
                verbose,
                create_strings,
                exon_mode )
```

Compares all alternative annotations given (alt_paths) to the reference annotation (ref_path) by calling annotation_sort and appends the results dictionaries to a list.

See also

annotation_sort()

Parameters

ref_path	Path to the reference annotation file
alt_path	Path to the alternative annotation file
verbose	If True, triggers display of more information messages. Default is 'False'
create_strings	Boolean indicating wether to use the 'old' comparison function (old_compare_loci, 'True') or the new one (compare_loci, 'False')
exon_mode	Boolean indicating if the main comparison structures read from the file should be coding sequences (CDS, False) or exons (True). Default is 'False' (CDS comparison)

See also

compare_loci()

old_compare_loci()

Returns

Returns the list of all results dictionaries of all alternatives

3.2.2.2 write_multi_results()

```
def CDSmulticompR.write_multi_results (
              multi_results,
              ref_path )
```

Writes the results returned by the function multicomp into a results synthesis CSV file detailing the loci identity for each alternative.

See also

multicomp()

Parameters

multi_results	List of results dictionaries, as returned by multicomp
ref_path	Path to the reference annotation GFF file

Chapter 4

Class Documentation

4.1 cluster.Cluster Class Reference

This class represents clusters of overlapping loci computed by the function construct_clusters.

Public Member Functions

- def __init__ (self, name, loci=None, end=-1)

  This method initialises the class.
- def get_loci (self)

  Retrieves the 'loci' attribute of the instance.
- def append_to_loci (self, annotation, value)

  Appends the given value to the list of the given annotation in the 'loci' attribute.
- def get_end (self)

  Retrieves the 'end' attribute of the instance.
- def set_end (self, value)

  Sets the given value as the 'end' attribute.
- def get_mRNAs (self, loc_id, ref)

  This method is used to retrieve the complete list of mRNAs of the locus of the cluster specified with its locus ID and a boolean indicating if the locus is to be searched for in the reference or alternative list.
- def get_details (self)

  Retrieves the class instance's 'loci' attribute as a human-readable dictionary of lists detailing each locus' mRNAs.

Public Attributes

- name

  name of the cluster
- loci

  list of Locus class instances present in the cluster
- end

  end coordinate of the cluster.

4.1.1 Detailed Description

This class represents clusters of overlapping loci computed by the function construct_clusters.

It posesses only three attributes: 'loci', which is a dictionary of lists of instances of the class 'Locus' indicating the loci attributed to each cluster; 'name'; and 'end' which is the end coordinate of the last loci of the cluster

See also

construct_clusters()

Locus

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def cluster.Cluster.__init__ (
              self,
              name,
              loci = None,
              end = -1 )
```

This method initialises the class.

Parameters

name	(optional) the name of the cluster
loci	(optional) the lists of loci for the cluster of the reference and alternative annotations, as a dictionary of lists
end	(optional) the end coordinate of the last locus of the cluster

4.1.3 Member Function Documentation

4.1.3.1 append_to_loci()

```
def cluster.Cluster.append_to_loci (
              self,
              annotation,
              value )
```

Appends the given value to the list of the given annotation in the 'loci' attribute.

Parameters

annotation	string ('ref' or 'alt') indicating the origin of the value
value	Value to append to the 'loci' attribute

4.1.3.2 get_details()

```
def cluster.Cluster.get_details (
              self )
```

Retrieves the class instance's 'loci' attribute as a human-readable dictionary of lists detailing each locus' mRNAs.

Returns

Returns a dictionary of list detailing the content of the instance's 'loci' attribute

4.1.3.3 get_end()

```
def cluster.Cluster.get_end (
              self )
```

Retrieves the 'end' attribute of the instance.

Returns

Returns the 'end' attribute of the class instance

4.1.3.4 get_loci()

```
def cluster.Cluster.get_loci (
              self )
```

Retrieves the 'loci' attribute of the instance.

Returns

Returns the 'loci' attribute of the class instance

4.1.3.5 get_mRNAs()

```
def cluster.Cluster.get_mRNAs (
              self,
              loc_id,
              ref )
```

This method is used to retrieve the complete list of mRNAs of the locus of the cluster specified with its locus ID and a boolean indicating if the locus is to be searched for in the reference or alternative list.

Parameters

loc↩ _id	Identifier of the locus from which to retrieve mRNAs
ref	Boolean indicating from which of the reference (True) or alternative (False) locus to retrieve the mRNAs

Returns

Returns the list of all mRNAs of the specified locus

Remarks

This method is intended to be used in unit tests to verify the clusters contains the intended loci

4.1.3.6 set_end()

```
def cluster.Cluster.set_end (
              self,
              value )
```

Sets the given value as the 'end' attribute.

Parameters

value	Value to give to the 'end' attribute of the class instance

4.1.4 Member Data Documentation

4.1.4.1 end

```
cluster.Cluster.end
```

end coordinate of the cluster.

Equal to the end coordinate of the last cluster's locus

The documentation for this class was generated from the following file:

- /home/runner/work/CDScompR/CDScompR/script/cluster.py

4.2 locus.Locus Class Reference

This class represents an annotation's locus identified from reading the associated GFF file with the get_gff_borders function.

Public Member Functions

- def __init__ (self, name="", mRNAs=None, start=-1, end=-1, direction="")

  This method initialises the class with default values for each attribute.
- def mRNAs (self)

  This method is used to retrieve the 'mRNAs' attribute of an instance.
- def set_mRNAs (self, value)

  Sets the value of the 'mRNAs' attribute of the class instance.
- def contain_mrnas (self, ∗∗mrnas)

  This method is used to verify if all mRNAs of a dictionary are present in the class instance's 'mRNAs' attribute.
- def reverse (self, cluster_end)

  Reverses the coordinates of all mRNAs of the class instance.
- def show_init (self)

  This function was added for convenience as a way to easily retrieve the attribute values of the class instance as a formatted string.

Public Attributes

- name

  name (identifier) of the locus
- mRNAs

  Dictionary of lists containing the CDS list of each mRNA of the locus.
- start

  start coordinate of the locus on the sequence
- end

  end coordinate of the locus on the sequence
- direction

  string indicating if the locus is on direct or reverse strand

4.2.1 Detailed Description

This class represents an annotation's locus identified from reading the associated GFF file with the get_gff_borders function.

It posesses multiple attributes describing the locus

See also

get_gff_borders()

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def locus.Locus.__init__ (
              self,
              name = "",
              mRNAs = None,
              start = -1,
              end = -1,
              direction = "" )
```

This method initialises the class with default values for each attribute.

Parameters

name	(optional) ID of the locus
mRNAs	(optional) Dictionary listing all mRNAs of the locus. Each key corresponds to the ID of the mRNA and each value to the list of all CDS coordinates as retrieved by the get_gff_borders function
start	(optional) Start coordinate of the locus
end	(optional) End coordinate of the locus
direction	(optional) String indicating if the locus is on the 'direct' or 'reverse' strand

See also

get_gff_borders()

4.2.3 Member Function Documentation

4.2.3.1 contain_mrnas()

```
def locus.Locus.contain_mrnas (
             self,
            ** mrnas )
```

This method is used to verify if all mRNAs of a dictionary are present in the class instance's 'mRNAs' attribute.

Parameters

∗∗*mrnas*	Dictionary with mRNA names as keys and CDS coordinates list as values

Returns

Returns a boolean indicating if all given mRNAs were found in the class instance's 'mRNAs' attribute

Remarks

This method is intended to be used in unit tests to verify the locus contains the intended mRNAs

4.2.3.2 mRNAs()

```
def locus.Locus.mRNAs (
              self )
```

This method is used to retrieve the 'mRNAs' attribute of an instance.

Returns

Returns the 'mRNAs' attribute of the class instance

4.2.3.3 reverse()

```
def locus.Locus.reverse (
              self,
              cluster_end )
```

Reverses the coordinates of all mRNAs of the class instance.

Parameters

cluster_end	End position coordinate of the cluster containing the locus

Remarks

This method doesn't return anything and modifies directly the class instance. This method is used in case of loci on the reverse strand, and transforms the coordinates into coordinates from the end of the parent cluster

4.2.3.4 set_mRNAs()

```
def locus.Locus.set_mRNAs (
              self,
              value )
```

Sets the value of the 'mRNAs' attribute of the class instance.

Parameters

value	Value to be assigned to the 'mRNAs' attribute

4.2.3.5 show_init()

```
def locus.Locus.show_init (
```

```
        self )
```

This function was added for convenience as a way to easily retrieve the attribute values of the class instance as a formatted string.

Returns

Returns a string describing the values of each attribute of the class instance

The documentation for this class was generated from the following file:

- /home/runner/work/CDScompR/CDScompR/script/locus.py

4.3 intervals_utils.OrderedIntervals Class Reference

This class represents a list of intervals and implements multiple basic operations between intervals.

Public Member Functions

- def __init__ (self, intervals, include_ub=False)

  Initialises the class instance with the given intervals.
- def get_intervals_with_included_ub (self)

  Returns the class instance's intervals list including the upper bound of each interval.
- def total_length (self)

  Calculates the total length of all intervals.
- def union (self, other)

  Calculates the union an interval list with another interval list.
- def intersection (self, other)

  Calculates the intersection of this interval list with another interval list.
- def difference (self, other)

  Calculates the difference of this interval list with another interval list and returns the intervals of this list (asymmetric difference)
- def symmetric_difference (self, other)

  Calculates the difference of this interval list with another interval list and returns both intervals list (symmetric difference)
- def merge (self, other, keep_operator)

  Merges this interval list with another interval list based on the specified operator.

Static Public Member Functions

- def transform_intervals_to_exclude_ub (intervals)

  Excludes the upper bound of each interval of the given list.
- def new (intervals, include_ub=False)

  Creates a new OrderedIntervals object.

Public Attributes

- intervals

  List of sequence coordinates representing CDS intervals.

4.3.1 Detailed Description

This class represents a list of intervals and implements multiple basic operations between intervals.

It is used to represent and compare lists of CDS bounds of compared loci in the mRNA comparison function compute_matches_mismatches_EI_RF

See also

compute_matches_mismatches_EI_RF()

Remarks

This class was adapted from https://stackoverflow.com/a/20062829 by Vincent Ranwez, and translated into python using chatGPT

4.3.2 Constructor & Destructor Documentation

4.3.2.1 __init__()

```
def intervals_utils.OrderedIntervals.__init__ (
              self,
              intervals,
              include_ub = False )
```

Initialises the class instance with the given intervals.

Parameters

intervals	List of interval bounds to represent
include_ub	Boolean indicating wether to include the upper bounds of each interval of the list

Remarks

for CDS bounds of GFF annotations, include_ub needs to be 'True'

4.3.3 Member Function Documentation

4.3.3.1 difference()

```
def intervals_utils.OrderedIntervals.difference (
              self,
              other )
```

Calculates the difference of this interval list with another interval list and returns the intervals of this list (asymmetric difference)

Parameters

other	Another OrderedIntervals object.

Returns

A new OrderedIntervals object representing the difference between the two interval lists.

Remarks

This method returns the intervals of the first interval list not present in the second, but not those of the second not present in the first. To get both, use the symmetric_difference method

See also

symmetric_difference()

4.3.3.2 get_intervals_with_included_ub()

```
def intervals_utils.OrderedIntervals.get_intervals_with_included_ub (
              self )
```

Returns the class instance's intervals list including the upper bound of each interval.

Returns

Returns the list of intervals of the class instance with their upper bounds

4.3.3.3 intersection()

```
def intervals_utils.OrderedIntervals.intersection (
              self,
              other )
```

Calculates the intersection of this interval list with another interval list.

Parameters

other	An OrderedIntervals object.

Returns

A new OrderedIntervals object representing the intersection of the two interval lists.

4.3.3.4 merge()

```
def intervals_utils.OrderedIntervals.merge (
              self,
              other,
              keep_operator )
```

Merges this interval list with another interval list based on the specified operator.

Parameters

other	Another OrderedIntervals object.
keep_operator	A function that determines whether to keep an interval based on its presence in either list.

Returns

A new OrderedIntervals object representing the merged intervals.

4.3.3.5 new()

```
def intervals_utils.OrderedIntervals.new (
              intervals,
              include_ub = False )  [static]
```

Creates a new OrderedIntervals object.

Parameters

intervals	List of integers representing the intervals.
include_ub	Boolean indicating if the upper bound should be included.

Returns

A new OrderedIntervals object.

4.3.3.6 symmetric_difference()

```
def intervals_utils.OrderedIntervals.symmetric_difference (
              self,
              other )
```

Calculates the difference of this interval list with another interval list and returns both intervals list (symmetric difference)

Parameters

other	Another OrderedIntervals object.

Returns

A new OrderedIntervals object representing the difference between the two interval lists.

Remarks

This method returns the intervals of the first interval list not present in the second, and those of the second not present in the first. To get only the difference for this instance's list, use the difference method

See also

difference()

4.3.3.7 total_length()

```
def intervals_utils.OrderedIntervals.total_length (
              self )
```

Calculates the total length of all intervals.

Returns

The sum of the lengths of all intervals.

4.3.3.8 transform_intervals_to_exclude_ub()

```
def intervals_utils.OrderedIntervals.transform_intervals_to_exclude_ub (
              intervals )  [static]
```

Excludes the upper bound of each interval of the given list.

Parameters

intervals	Interval list to be transformed

Returns

Returns the same list of intervals with each upper bound removed

4.3.3.9 union()

```
def intervals_utils.OrderedIntervals.union (
              self,
              other )
```

Calculates the union an interval list with another interval list.

Parameters

other	An OrderedIntervals object.

Returns

A new OrderedIntervals object representing the union of the two interval lists.

The documentation for this class was generated from the following file:

- /home/runner/work/CDScompR/CDScompR/script/intervals_utils.py

Index