Package 'AssessORF'

April 15, 2019

Type Package

Title Assess Gene Predictions Using Proteomics and Evolutionary Conservation

Version 1.0.2

Date 2018-11-8

Description In order to assess the quality of a set of predicted genes for a genome, evidence must first be mapped to that genome. Next, each gene must be categorized based on how strong the evidence is for or against that gene. The AssessORF package provides the functions and class structures necessary for accomplishing those tasks, using proteomic hits and evolutionarily conserved start codons as the forms of evidence.

Depends R (>= 3.5.0), DECIPHER (>= 2.8.0)

Imports Biostrings, GenomicRanges, IRanges, graphics, grDevices, methods, stats, utils

Suggests AssessORFData, BiocStyle, knitr, rmarkdown

biocViews ComparativeGenomics, GenePrediction, GenomeAnnotation, Genetics, Proteomics, QualityControl, Visualization

License GPL-3

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 6.1.0

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/AssessORF

git_branch RELEASE_3_8

git_last_commit 670115f

git_last_commit_date 2018-11-08

Date/Publication 2019-04-15

Author Deepank Korandla [aut, cre], Erik Wright [aut]

Maintainer Deepank Korandla <dkorandl@andrew.cmu.edu>

R topics documented:

`as.matrix.Assessment` *Tabulate the Category Assignments for Assessment Results Objects*

Description

The `as.matrix` method for `Assessment` and subclass `Results` objects

Usage

```
## S3 method for class 'Assessment'
as.matrix(x, ...)
```

Arguments

`x`	An object of class `Assessment` and subclass `Results`.
`...`	Additional arguments.

Details

`as.matrix.Assessment` tabulates and returns the number of times each category appears in the `CategoryAssignments` vector within the given `Results` object. If the number of genes for any the 12 main gene / ORF categories is zero, a count (of zero) will still be included for that category.

Value

A one-row matrix with the counts for the number of genes/ORFs that fall into each category. The corresponding category codes serve as the column names, and the name of the row is the strain ID.

See Also

`Assessment-class`

Examples

```
as.matrix(readRDS(system.file("extdata",
                              "MGAS5005_PreSaved_ResultsObj_Prodigal.rds",
                              package = "AssessORF")))
```

AssessGenes *Assess Genes*

Description

Assess and categorize a set of genes for a genome using proteomics hits, evolutionarily conserved starts, and evolutionarily conserved stops as evidence

Usage

```
AssessGenes(geneLeftPos,
            geneRightPos = NA_integer_,
            geneStrand = NA_character_,
            inputMapObj,
            geneSource = "",
            minCovNum = 10,
            minCovPct = 5,
            minConCovRatio_Best = 0.99,
            limConCovRatio_NotCon = 0.8,
            minConCovRatio_Stop = 0.5,
            noConStopsGeneFrac = 0.5,
            minNumStops = 2,
            minMissORFLen = 0,
            allowNestedORFs = FALSE,
            useNTermProt = FALSE,
            verbose = TRUE)
```

Arguments

geneLeftPos	An integer vector with the left positions of each gene, in terms of the forward strand. Can also be a GRanges object from the GenomicRanges package that holds all of the positional information (including strand) for the genes. In that case, the next two parameters should be left as NA.
geneRightPos	An integer vector with the right positions of each gene, in terms of the forward strand. Should be left at the default value of NA_integer_ if geneLeftPos is a GRanges object.
geneStrand	A character vector consisting of "+" and "-", specifying which strand each gene is on. Should be left at the default value of NA_character_ if geneLeftPos is a GRanges object.
inputMapObj	EITHER an object of class Assessment and subclass DataMap OR a character string corresponding to the strain identifier for one of such objects from AssessORFData.
geneSource	Optional character string that describes the source of the gene set, i.e. a database or gene prediction program. Used when viewing and identifying the object returned by the function.
minCovNum	Minimum number of related genomes required to have synteny to a position in the central genome. Recommended to use the default value.
minCovPct	Minimum percentage of related genomes required to have synteny to a position in the central genome. Must be an integer ranging from 0 to 100. Recommended to use the default value.

`minConCovRatio_Best`
: Minimum value of the start codon conservation to coverage ratio needed to call a start conserved. Must range from 0 to 1. Lower values allow more conserved starts through. Recommended to use the default value.

`limConCovRatio_NotCon`
: Maximum, non-inclusive value of the conservation to coverage ratio needed to call a possible conserved start not conserved. Used when making a decision on how to categorize the conserved start evidence. Must range from 0 to 1 Recommended to use the default value.

`minConCovRatio_Stop`
: Minimum value of the stop codon conservation to coverage ratio needed to say a position in the central genome corresponds to a conserved stop across the related genomes. Must range from 0 to 1. Lower values allow more conserved stops through. Recommended to use the default value.

`noConStopsGeneFrac`
: Value from 0 to 1 describing the fractional range of positions in a gene, starting from the start of the gene and moving towards the stop of the gene, to use in searching for conserved stops. For example, a value of 0.25 means that the first quarter of the gene is checked for conserved stops, a value of 0.5 correspond to the first half of the gene, etc. Recommended to use the default value.

`minNumStops`
: Minimum number of conserved stop positions required to be within a gene (with no mapped proteomics hits) in order to categorize that gene as an overprediction. Recommended to use the default value.

`minMissORFLen`
: Minimum ORF length required to include an ORF with protein hits but no gene start in the final results.

`allowNestedORFs`
: Logical indicating whether or not to include ORFs with protein hits but no gene starts that are completely nested within an ORF in another frame in the final results.

`useNTermProt`
: Logical indicating whether or not to require the given gene starts to align with (or be one codon off from) the start of the first proteomics hit in the ORF. The mapping object must be built with N-terminal proteomics data. Default value is FALSE.

`verbose`
: Logical indicating whether or not to display progress and status messages.

Details

For each of the given genes, `AssessGene` assigns a category based on where conserved starts, conserved stops, and/or proteomics hits are located in relation to the start of the gene. The category assignments for the genes are stored in the `CategoryAssignments` vector in the `Results` object returned by the function. Please see `Assessment-class` for a list of all possible categories and their descriptions.

If `geneLeftPos` is a `GRanges` object, then the left and right positions of each gene along with the strand of each gene are extracted from the object. Any sequence names given for the genes within the `GRanges` object are ignored, and the `CategoryAssignments` in the returned `Results` object follows the same order as to how the genes are listed within the `GRanges` object.

If gene positional information is instead given as three vectors, then the three vectors, `geneLeftPos`, `geneRightPos`, and `geneStrand`, must all be of the same length. Using the same index with each vector must provide information on the same gene (think of the vectors as columns of the same table). `geneLeftPos` and `geneRightPos` describe the upstream and downstream positions (respectively) for each gene in terms of the forward strand. For genes on the forward strand, `geneLeftPos`

corresponds to the start positions and geneRightPos corresponds to stop positions. For genes on the reverse strand, geneLeftPos corresponds to the stop positions and geneRightPos corresponds to the start positions. Gene positions on the reverse strand must be relative to the 5' to 3' direction of the forward strand (as opposed to being relative to the 5' to 3' direction of the reverse strand). This means that none of the elements of geneLeftPos can be greater than (or equal to) the corresponding element in geneRightPos. The CategoryAssignments in the returned Results object has the same length as and aligns with the indexing of the three given gene positional information vectors.

Please ensure that the same genome used in the mapping function is also used to derive the set of genes for this assessment function. The function will only error if any gene positions are outside the bounds of the genome and does not make any other checks to make sure the genes are valid for the genome.

The maximum of either minCovNum or (minCovPct/100) multiplied by the number of related genomes is used as the minimum coverage required in determining conserved starts and stops.

Additionaly, open reading frames with proteomics evidence but no gene start are categorized based on whether or not there is a conserved start upstream of the proteomic evidence. The positions and lengths of these open reading frames is included in the N_CS-_PE+_ORFs and N_CS+_PE+_ORFs matrices within the final object that is returned.

Value

An object of class Assessment and subclass Results

See Also

Assessment-class

Examples

```
## Example showing the minimum number of arguments that need to be specified:

## Not run:
myResObj <- AssessGenes(geneLeftPos = myGenesLeft,
                        geneRightPos = myGenesRight,
                        geneStrand = myGenesStrand,
                        inputMapObj = myMapObj)

## End(Not run)



## Example from vignette is shown below

currMapObj <- readRDS(system.file("extdata",
                                  "MGAS5005_PreSaved_DataMapObj.rds",
                                  package = "AssessORF"))

currProdigal <- readLines(system.file("extdata",
                                      "MGAS5005_Prodigal.sco",
                                      package = "AssessORF"))[-1:-2]

prodigalLeft <- as.numeric(sapply(strsplit(currProdigal, "_", fixed=TRUE), `[`, 2L))
prodigalRight <- as.numeric(sapply(strsplit(currProdigal, "_", fixed=TRUE), `[`, 3L))
```

```
prodigalStrand <- sapply(strsplit(currProdigal, "_", fixed=TRUE), `[`, 4L)

currResObj <- AssessGenes(geneLeftPos = prodigalLeft,
                          geneRightPos = prodigalRight,
                          geneStrand = prodigalStrand,
                          inputMapObj = currMapObj,
                          geneSource = "Prodigal")
```

`Assessment` *Assessment objects*

Description

In order to assess the quality of a set of (predicted) genes for a genome, evidence must first be mapped to that genome. Next, each gene must be categorized based on how strong the evidence is for that gene or against that gene. Class `Assessment` furnishes objects that can store the neccesary information for assessing a set of genes for a genome and also provides functions for viewing and visualizing assessment information. Specifically, class `Assessment` objects utilize proteomic hits and evolutionarily conserved start & stop codons as evidence to determine the correctness for each gene in a given set.

`DataMap` Objects

Objects of class `Assessment` and subclass `DataMap` are used to store the mapping of proteomics and evolutionary conservation to the genome of interest (central genome). They are generated through the function `MapAssessmentData`, and they have a list structure containing the following elements:

`StrainID` Equal to `strainID` if it was specified; otherwise ""

`Species` Equal to `speciesName` if it was specified; otherwise ""

`GenomeLength` Length of the central genome

`StopsByFrame` Where the stops are in each frame, used to bound open reading frames in downstream functions

`N-TermProteomics` Logical describing whether or not the proteomics hits are from N-terminal proteomics

`FwdProtHits` Proteomic hit information that maps to the three forward frames of the central genome

`RevProtHits` Proteomic hit information that maps to the three reverse frames of the central genome

`FwdCoverage` Coverage of the forward strand of the central genome

`FwdConStarts` Start codon conservation of the forward strand of the central genome

`FwdConStops` Stop codon conservation of the forward strand of the central genome

`RevCoverage` Coverage of the reverse strand of the central genome

`RevConStarts` Start codon conservation of the reverse strand of the central genome

`RevConStops` Stop codon conservation of the reverse strand of the central genome

`NumRelatedGenomes` Number of related genomes that were mapped to the central genome

`HasProteomics` Logical describing whether or not proteomics evidence has been mapped to the central genome

`HasConservation` Logical describing whether or not evolutionary conservation evidence has been mapped to the central genome

Results Objects

Objects of class Assessment and subclass Results are used to store how correct a set of genes for a given genome. The function AssessGenes generates Results using a DataMap object and information on a set of genes for the genome corresponding to the DataMap object. Results objects have a list structure containing the following elements:

StrainID Equal to the strainID of the corresponding DataMap object

Species Equal to speciesName of the corresponding DataMap object

GenomeLength Length of the genome

GeneLeftPos Left positions of the given set of genes (in forward strand terms)

GeneRightPos Right positions of the given set of genes (in forward strand terms)

GeneStrand Strand information of the given set of genes ("+" or "-")

GeneSource The source of the given set of genes

NumGenes Number of genes given

N_CS-_PE+_ORFs Data for open reading frames with no gene start but with proteomics evidence

N_CS+_PE+_ORFs Data for open reading frames with no gene start but with proteomics evidence and at least one valid evolutionarily conserved start

CategoryAssignments A character vector that stores the category assignment for each of the given genes in the same order as the gene information (please see below for a list of all possible categories, their descriptions, and their character string codes)

Gene Categories

The CategoryAssignments vector in Results objects describes how the proteomics evidence and evolutionarily conserved start/stop codon evidence support or disprove the corresponding set of genes. In the vector, each gene is assigned a character string code that has the following format: "Y CS[_] PE[_]". The first part, the "Y", signifies that for this ORF contains a predicted gene. The second part, the "CS[_]", describes how the conserved start(s) lines up with the given gene start. The third part, the "PE[_]", describes how the proteomics hits line up with the given gene start.

Y CS+ PE+ There is a good conserved start aligned with the gene start with protein evidence downstream.

Y CS+ PE- There is a good conserved start aligned with the gene start without protein evidence downstream.

Y CS- PE+ There is no good conserved start aligned with the predicted start, and there is protein evidence downstream of the gene start.

Y CS- PE- There is no good conserved start aligned with the predicted start, and there is no protein evidence downstream of the gene start.

Y CS! PE- There are either multiple good conserved stops in the middle of the gene, or the most downstream, good conserved stop is followed by a good conserved start. There is no protein evidence downstream of the gene start

Y CS! PE+ The most downstream, good conserved stop is followed by a good conserved start, and there is protein evidence downstream of the gene start.

Y CS< PE! The protein evidence disagrees with/is upstream of the gene start, and there is a good conserved start upstream of the protein evidence.

Y CS- PE! The protein evidence disagrees with/is upstream of the gene start, and there is no good conserved start upstream of the protein evidence.

`Y CS> PE+` The best conserved starts are downstream of the predicted start, and there is protein evidence downstream of the gene start.

`Y CS> PE-` The best conserved starts are downstream of the predicted start, and there is no protein evidence downstream of the gene start.

`Y CS< PE+` At least one of the best conserved starts is upstream of the predicted start, and there is protein evidence downstream of the gene start.

`Y CS< PE-` At least one of the best conserved starts is upstream of the predicted start, and there is no protein evidence downstream of the gene start.

S3 Methods

`as.matrix.Assessment` (only works with objects of class `Results`)

`print.Assessment`

`plot.Assessment`

`mosaicplot.Assessment` (only works with objects of class `Results`)

`MapAssessmentData` *Map Evidence to a Genome*

Description

Maps proteomics hits and evolutionarily conserved starts to a central genome

Usage

```
MapAssessmentData(genomes_DBFile,
                  tblName = "Seqs",
                  central_ID,
                  related_IDs,
                  protHits_Seqs,
                  protHits_Scores = rep.int(1, length(protHits_Seqs)),
                  strainID = "",
                  speciesName = "",
                  protHits_Threshold = 0,
                  protHits_IsNTerm = FALSE,
                  related_KMerLen = 8,
                  related_MinDist = 0.01,
                  related_MaxDistantN = 1000,
                  startCodons = c("ATG", "GTG", "TTG"),
                  useProt = TRUE,
                  useCons = TRUE,
                  verbose = TRUE)
```

Arguments

`genomes_DBFile`	A SQLite connection object or a character string specifying the path to the database file.
`tblName`	Character string specifying the table where the genome sequences are located.

`central_ID` Character string specifying which identifier corresponds to the central genome, the genome to which the proteomics data and evolutionary conservation data will be mapped.

`related_IDs` Character vector of strings specifying identifiers that correspond to related genomes, the genomes that will be used to determine which start codons (ATG, GTG, and TTG) are evolutionarily conserved.

`protHits_Seqs` Character vector of amino acid strings that correspond to the sequences for the proteomics hits.

`protHits_Scores`
Numeric vector of (confidence) scores for the proteomics hits. Scores cannot be negative. The default option assigns a score of one to each proteomics hit.

`strainID` Optional character string that specifies the strain identifier that the central genome corresponds to.

`speciesName` Optional character string that specifies the name of the species that the central genome corresponds to.

`protHits_Threshold`
Optional number that specifies what percent of the lowest scoring proteomics hits should be dropped. Must be a non-negative integer less than 100.

`protHits_IsNTerm`
Logical describing whether or not the proteomics hits come from N-terminal proteomics. Default value is false.

`related_KMerLen`
The k-mer length to be used when measuring distances between the central genome and related genomes. Default value is 8. Recommended to use the default value.

`related_MinDist`
The minimum fractional distance required for a related genome to be used in finding evolutionary conservation. Used to prevent the inclusion of related genomes that are too similar to the central genome. Default value is 0.01. Recommended to use the default value.

`related_MaxDistantN`
The maximum number of related genomes to use in finding evolutionary conservation after the related genomes have been sorted from most distantly related to most closely related in relation to the central genome. Default value is 1000.

`startCodons` A charcter vector consisting of three-letter DNA strings to use as the start codons when finding evolutionarily conserved starts.

`useProt` Logical indicating whether or not proteomics evidence should be mapped to the genome. Default value is true. Cannot be false if `useCons` is false.

`useCons` Logical indicating whether or not evolutionary conservation evidence should be mapped to the genome. Default value is true. Cannot be false if `useProt` is false.

`verbose` Logical indicating whether or not to display progress and status messages.

Details

`MapAssessmentData` maps the given data (either proteomics data, evolutionary conservation data, or both) to the given central genome and stores those mappings in the object outputted by the function. The object that is outputted can then be used to assess the quality of genes predicted for that same central genome.

All genomes used inside this function, including the central genome, must be inside the specified table of the specified database. If the central genome is not found, the function returns an error. Please see the Using AssessORF vignette for details on how to populate a database with genomic sequences.

Information on the proteomics hits is primarily given by `protHits_Seqs` and `protHits_Scores`. The sequences (`protHits_Seqs`) are mapped to the six-frame translations of the central genome, and the scores (`protHits_Scores`) are used in thresholding and plotting the proteomics hits.

`protHits_Scores` can be a single number. In that case, that number is used the as the score for all proteomics hits. Otherwise, the `protHits_Scores` must be of the same length as `protHits_Seqs`.

Only proteomics hits with a score greater than the value of the percentile that corresponds to the value of `protHits_Threshold` will be kept and the rest of the hits will be dropped. If all the proteomics hits have the same score or if `protHits_Threshold` is zero, no thresholding will occur and no hits will be dropped.

Please note that `protHits_IsNTerm` has no affect on how the proteomics evidence is mapped to the central genome but it can be used to affect how genes are assessed and categorized in `AssessGenes`.

Evolutionarily conserved starts and conserved stop are found by first measuring how far the related genomes are from the central genome using k-mer frequencies. Next, the most distant related genomes are aligned to the central genome. This provides information on how often each position in the central genome is covered by syntenic matches to related genomes (coverage), how often those positions correspond to the start codons (start codon conservation) in both genomes, and how often those positions correspond to stop codons in related genomes (stop codon conservation). A ratio of conservation to coverage is used in downstream functions to measure the strength of both conserved starts and conserved stops.

Related genomes should be from species that are closely related to the given strain. `related_IDs` specifies the identifiers for the sequences of the related genomes inside the database. A related genome identifier (each element of `related_IDs`) is considered invalid and not used when finding evolutionary conservation if it is not found in the databse. Please note that the function will only error when none of the related genomes are found.

If there are less valid related genomes in the sequence database than value of `related_MaxDistantN`, all valid related genomes will be used in finding evolutionary conservation.

The logical flag `useProt` is used to indicate whether or not proteomics evidence has been provided and should be mapped to the genome. Error checking will not occur for any arguments that involve proteomics if it is false.

The logical flag `useCons` is used to indicate whether or not evolutionary conservation evidence has been provided and should be mapped to the genome. Error checking will not occur for any arguments that involve evolutionary conservation if it is false.

Value

An object of class `Assessment` and subclass `DataMap`

See Also

`Assessment-class`

Examples

```
## Example showing the minimum number of arguments that need to be specified
## to map both proteomics and evolutionary conservation data:
```

```
## Not run:
myMapObj <- MapAssessmentData(myDBFile, central_ID = "1",
                              related_IDs = as.character(2:1001),
                              protHits_Seqs = myProtSeqs)

## End(Not run)


## Runnable example that uses evolutionary conservation data only:
## Human adenovirus 1 is the strain of interest, and the set of Adenoviridae
## genomes will serve as the set of genome. The cenral genome, also known as
## the genome of human adenovirus 1, is at identifier 1. The related genomes
## are at identifiers 2 - 13.

myMapObj <- MapAssessmentData(system.file("extdata",
                                          "Adenoviridae.sqlite",
                                          package = "AssessORF"),
                              central_ID = "1",
                              related_IDs = as.character(2:13),
                              speciesName = "Human adenovirus 1",
                              useProt = FALSE)
```

`mosaicplot.Assessment` *Plot Genes by Category and Length*

Description

The `mosaicplot` method for `Assessment` object

Usage

```
## S3 method for class 'Assessment'
mosaicplot(x, ...)
```

Arguments

`x`	An object of class `Assessment` and subclass `Results`.
`...`	Further `mosaicplot` parameters.

Details

`mosaicplot.Assessment` plots all the genes in the given `Results` object by category and length. This set of genes includes both the supplied predicted genes as well as open reading frames with proteomics evidence but no predicted start.

The set of genes are separated into ten quantile bins based on the length of the gene/open reading frame. The genes are then plotted by length bin and category in a mosaic format, with each column representing a length bin and each row/block representing a category.

Value

Invisibly returns the input object `x`

See Also

Assessment-class

Examples

```
mosaicplot(readRDS(system.file("extdata",
                               "MGAS5005_PreSaved_ResultsObj_Prodigal.rds",
                               package = "AssessORF")))
```

---

plot.Assessment *Plot Assessment Objects*

Description

The plot method for Assessment objects

Usage

```
## S3 method for class 'Assessment'
plot(x, y = NULL, related_MinConStart = 0.8, ...)
```

Arguments

x	An object of class Assessment and of either subclass DataMap or subclass Results.
y	An optional object of class Assessment and of either subclass DataMap or subclass Results. Its subclass must be different than the subclass of x
related_MinConStart	Minimum value of the conservation to coverage ratio needed to call a start conserved. Must range from 0 to 1. Lower values allow more conserved starts through. Recommended to use default value.
...	Further plotting parameters.

Details

If only x is specified and x is of subclass DataMap, an interactive genome viewer showing how the proteomics data and evolutionary conservation data maps to the central genome is plotted.

If only x is specified and x is of subclass Results, a bar chart describing the number of genes in each category is plotted.

If both x and y are specified, an interactive genome viewer showing how the proteomics data, evolutionary evolutionary conservation data, and gene set map to the central genome is plotted.

Value

Invisibly returns the input object x

See Also

Assessment-class, locator

Examples

```
currMapObj <- readRDS(system.file("extdata",
                                  "MGAS5005_PreSaved_DataMapObj.rds",
                                  package = "AssessORF"))

currResObj <- readRDS(system.file("extdata",
                                  "MGAS5005_PreSaved_ResultsObj_Prodigal.rds",
                                  package = "AssessORF"))

plot(currMapObj)

plot(currResObj)

plot(currMapObj, currResObj)

plot(currResObj, currMapObj)
```

---

print.Assessment *Print Assessment Objects*

Description

The `print` method for `Assessment` objects

Usage

```
## S3 method for class 'Assessment'
print(x, ...)
```

Arguments

`x`	An object of class `Assessment` and of either subclass `DataMap` or subclass `Results`.
`...`	Further printing parameters.

Details

If `x` is of subclass `DataMap`, the length of the genome is printed along with any supplied identifying information for the genome.

If `x` is of subclass `Results`, the number of genes in each category and the accuracy scores are printed out along with any supplied identifying information.

Value

Invisibly returns the input object `x`

See Also

`Assessment-class`

Examples

```
print(readRDS(system.file("extdata",
                          "MGAS5005_PreSaved_DataMapObj.rds",
                          package = "AssessORF")))

print(readRDS(system.file("extdata",
                          "MGAS5005_PreSaved_ResultsObj_Prodigal.rds",
                          package = "AssessORF")))
```

`ScoreAssessmentResults` *Score Gene Assessment Results*

Description

Scores the results from the assessment of a set of genes using one of three modes

Usage

```
ScoreAssessmentResults(x, mode = "a")
```

Arguments

`x`	An object of class `Assessment` and subclass `Results`.
`mode`	Must either be "a" (use all evidence), "p" (use proteomics evidence only), "c" (use evolutionary conservation evidence only), or "w" (use all evidence but with weights).

Details

`ScoreAssessmentResults` calculates an accuracy-like score for the categorization of genes within the given `Results` object using the given mode of calculation. The score for a mode is equal to the number of genes that were categorized to be correct for that mode divided by the total number of genes that could have been categorized as correct for that mode (i.e. a count of the number of genes that had available and useable evidence for that particular mode).

Open reading frames with proteomics evidence but no predicted start are included in the total gene count when calculating the accuracy-like score for the proteomics mode and for both all evidence modes.

In the weighted, all evidence mode, weights for each category are determined by the number of types of evidence that are supporting or against genes in the category. Counts are multiplied by their corresponding weight, and the maximum value for a weight is 2.

Value

A numeric vector of length one containing the calculated accuracy-like score.

See Also

`Assessment-class`

Examples

```
currResObj <- readRDS(system.file("extdata",
                                  "MGAS5005_PreSaved_ResultsObj_Prodigal.rds",
                                  package = "AssessORF"))

ScoreAssessmentResults(currResObj, "a")

ScoreAssessmentResults(currResObj, "c")

ScoreAssessmentResults(currResObj, "p")
```

Index