Functions.py

Contains all the functions needed to analyze the datasets.

Classes:

DataTables

class **DataTables ():**
The interface that defines the abstract methods for the class DiseaseTable and GeneTable.

Analysis

class **Analysis ():**
The interface that defines the abstract methods for the class Testing.

DiseaseTable

class **DiseaseTable (** *table, delimiter=None,* **):**
The class creates the DiseaseTable with the parameter table.

Parameters:

table : ***pandas.DataFrame***
the tsv file containing the table.

delimiter : ***string or none***
Set the delimiter based on the extension (.tsv or .csv).

Attributes:

__diseaseTable : ***pandas.DataFrame***
(private) The disease dataframe used by the class methods.

Methods:

__getitem__ (*item,* **): --> pandas.DataFrame**
Allows the use of slicing on the instance of the class. It return the data table sliced by index(es).

Method parameters:
item : ***slice***
the index of the row(s) to use for slicing.

get_table (): --> pandas.DataFrame
The function returns the table.

get_dimensions (): --> tuple
The function records the number of rows and column of dataframe.

get_labels (): --> list
The function records the labels of each columns of a dataframe.

get_head (): --> pandas.DataFrame
The function returns the first ten rows of the dataframe.

get_tail (): --> pandas.DataFrame
The function returns the last ten rows of the dataframe.

distinct (): --> pandas.DataFrame
It returns a dataframe of unique diseases (disease_name, diseaseid) present in the dataframe.
Every word of the diseases is capitalized to allow the sorting algorithm to sort them correctly instead of putting the lowercase at the end.

evidence (*disease,* **): --> pandas.DataFrame**
Receives as input a diseaseID or a disease name and returns a dataframe with the sentences that relates the COVID-19 with the disease.

Method parameters:
disease : ***str***
the diseaseID or disease name input.

GeneTable

class **GeneTable (** *table, delimiter=None,* **):**
The class creates the GeneTable with the parameter table.

Parameters:

table : ***pandas.DataFrame***
the tsv file containing the table.

delimiter : ***string or none***
Set the delimiter based on the extension (.tsv or .csv).

Attributes:

__geneTable : ***pandas.DataFrame***
(private) The gene dataframe used by the class methods.

Methods:

__getitem__ (*item,* **): --> pandas.DataFrame**
Allows the use of slicing on the instance of the class and returns the data table sliced by index(es).

Method parameters:
item : ***slice***
the index of the row(s) to use for slicing.

get_table (): --> pandas.DataFrame
The function returns the table.

get_dimensions (): --> tuple
The function records the number of rows and column of dataframe.

get_labels (): --> list
The function records the labels of each columns of a dataframe.

get_head (): --> pandas.DataFrame
The function returns the first ten rows of the dataframe.

get_tail (): --> pandas.DataFrame
The function returns the last ten rows of the dataframe.

distinct (): --> pandas.DataFrame
It returns a dataframe of unique genes (gene_symbol, geneid) present in the dataframe.

evidence (*gene,* **): --> pandas.DataFrame**
Receives as input a geneID or a gene symbol and returns a dataframe with the sentences that relates the COVID-19 with the gene.

Method parameters:
gene : ***str***
the geneID or gene symbol input.

Testing

class **Testing (** *geneTable, diseaseTable, geneDelimiter=None, diseaseDelimiter=None,* **):**
The class obtain both dataframes to find associations and relations between the element of the datasets.

Parameters:

geneTable : ***pandas.DataFrame***
the tsv file containing the genetable.

diseaseTable : ***pandas.DataFrame***
the tsv file containing the diseasetable.

geneDelimiter : ***string or none***
Set the delimiter based on the extension (.tsv or .csv).

diseaseDelimiter : ***string or none***
Set the delimiter based on the extension (.tsv or .csv).

Attributes:

__diseaseTable : ***pandas.DataFrame***
(private) The disease dataframe used by the class methods.

__geneTable : ***pandas.DataFrame***
(private) The gene dataframe used by the class methods.

Methods:

correlation_gene_disease (): --> pandas.DataFrame
The function returns a dataframe with the correlation between genes and diseases sorted by the most frequent.

Steps:
1) Merging of the two dataframes:
The merge occurs on pmid and nsentence (instead of sentence) because they are interchangeable as in the same publication the nth sentence ('nsentence') will always be 'sentence'. But in this way the program runs faster because it has to check only some numbers instead of whole strings to know which rows to merge.
2) Dropping duplicates:
The same concept goes for 'drop_duplicates'. When the function drop_duplicates() search for duplicates of the subset, with 'nsentence' it avoids checking for whole strings as it would instead do with sentences. 'geneid' and 'diseaseid' follow the same concept and are used instead of 'gene_symbol' and 'disease_name'.
3) Keeping only the columns needed, thus one for gene and one for disease
4) Count occurrences of the couple gene-disease and create a new dataframe with a couple as row and their occurrences in a new column; labels: ['gene_symbol', 'disease_name', 'occurrences'].

find_diseases_related_to_gene (*gene,* **): --> pandas.DataFrame**
The function receive as input a geneID or a gene symbol and then returns a dataframe with the diseases related to the gene.

Steps:
1) Merging of the two dataframes:
The merge occurs on pmid and nsentence (instead of sentence) because they are interchangeable as in the same publication the nth sentence ('nsentence') will always be 'sentence'. But in this way the program runs faster because it has to check only some numbers instead of whole strings to know which rows to merge.
2) Dropping duplicates:
The same concept goes for 'drop_duplicates'. When the function drop_duplicates() search for duplicates of the subset, with 'nsentence' it avoids checking for whole strings as it would instead do with sentences. 'geneid' and 'diseaseid' follow the same concept and are used instead of 'gene_symbol' and 'disease_name'.
3) Performing search:
It first tries to convert gene (string) given as input to an int. If it can, then it means it's a genid and only only the rows whose value in the columns 'geneid' will be 'gene' will be kept. Otherwise it means 'gene' given as input is a 'gene_symbol' and only the rows whose value in the columns 'gene_symbol' will be 'gene' given as input will be kept.
4) Keeping only the columns needed
5) Using title() on all diseases:
Some diseases are all lowercase and when sorted will be placed at the end of the table as the majority have the first letter uppercase. With title() all words of the diseases are now capitalized, and sort_values() will do what we want.
6) Dropping duplicates and sorting.

Method parameters:
gene : ***str or int***
the geneid or gene_symbol input.

find_genes_related_to_disease (*disease,* **): --> pandas.DataFrame**
The function receive as input a diseaseid or a disease_name and then returns a dataframe with the diseases related to the gene.

Steps:
1) Merging of the two dataframes:
The merge occurs on pmid and nsentence (instead of sentence) because they are interchangeable as in the same publication the nth sentence ('nsentence') will always be 'sentence'. But in this way the program runs faster because it has to check only some numbers instead of whole strings to know which rows to merge.
2) Dropping duplicates:
The same concept goes for 'drop_duplicates'. When the function drop_duplicates() search for duplicates of the subset, with 'nsentence' it avoids checking for whole strings as it would instead do with sentences. 'geneid' and 'diseaseid' follow the same concept and are used instead of 'gene_symbol' and 'disease_name'.
3) Performing search:
Check if the disease matches a pattern which consist of the first element as a 'C' and then at least 7 numbers until the end of the string. If it matches, then it means it's a 'diseaseid' and only the rows whose value in the columns 'diseaseid' are 'disease' will be kept. Otherwise it means 'disease' given as input is a 'disease_name' and only the rows whose value in the columns 'disease_name' are the 'gene' given as input will be kept.
4) Keeping only the columns needed.
5) Dropping duplicates and sorting.

Method parameters:
disease : ***str***
the diseaseid or disease_name input.