SoftwareX 12 (2020) 100526

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

GSimPy: A Python package for measuring group similarity

Yifei Zhang [a,b], Jia Cao [a,b,*]

[a] School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China
[b] Engineering Research Center for Forestry-oriented Intelligent Information Processing, National Forestry and Grassland Administration, Beijing 100083, China

ARTICLE INFO

Article history:
Received 16 November 2019
Received in revised form 18 May 2020
Accepted 20 May 2020

Keywords:
Group similarity
Group–Item–Element Model
Python package

ABSTRACT

In this paper, we present an open source Python package—GSimPy, which can measure similarity between items and groups that satisfy GIE model. GIE model is a Group–Item–Element model commonly used in many fields such as biology. GSimPy can help manage scientists' customized data and provide a random model as a reference to evaluate similarity results. Visualization tools are also supported to make analysis results more interpretable.

Code metadata

Current code version	0.0.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_354
Legal code license	MIT License
Code version system used	git
Software code languages, tools, and services used	Python 3.7
Compilation requirements, operating environments & dependencies	Tested on Mac OS X. Should work on Linux, Windows and Mac OS. Depends on NumPy, Matplotlib, and pyecharts.
If available Link to developer documentation/manual	https://github.com/curlyae1995/GSimPy
Support email for questions	curlyae1995@bjfu.edu.cn

1. Motivation and significance

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters) [1]. With the diversity in both definition of cluster and the strategy of clustering, lots of clustering algorithms appeared in literature, such as partitioning methods, hierarchical methods, density-based methods, model-based methods and grid-based methods [2]. Different clustering algorithms can divide data into different groups. The quantitative measurement of similarity between groups raises more and more attention. Understanding the associations between groups not only helps scientists discover some new scientific features at group level, but also becomes a new indicator to measure clustering algorithms in turn.

Here, we define a generic Group–Item–Element (GIE) model, that is, an item can be annotated by a set of 'atomic' elements, and a group is composed of many items. A simple item example is given in Fig. 1(a), where the items i and j are annotated by element sets (e_2, e_3, e_4) and (e_1, e_2, e_3) respectively. The relationship between two items can be determined by three scales, which are the element number of item i, the element number of item j, and the shared elements number between items i and j. As far as the three scales are determined, the similarity between two items can be calculated. Many methods using set-based or vector-based strategy to measure similarity between two items, such as Jaccard Index [3] and Cosine similarity [4], both are widely used in many scientific fields. Fig. 1(b) depicts a group example where group I and J are composed of items (i_1, i_2) and (j_1, j_2) respectively. When it comes to group, the methods for calculating similarity are mostly based on the similarity matrix. Some similarity methods consider every pairwise combination of items from two groups (all pairs technique), while others consider only the best-matching pair (best pairs technique) [5].

* Corresponding author.
E-mail addresses: curlyae1995@bjfu.edu.cn (Y. Zhang), caojia@bjfu.edu.cn (J. Cao).

https://doi.org/10.1016/j.softx.2020.100526

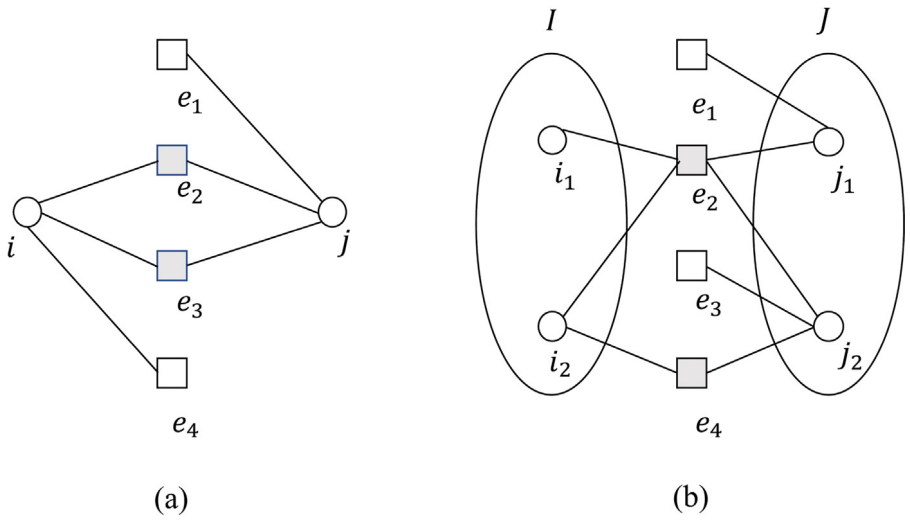


Fig. 1. Two Group–Item–Element examples. (a) The relationship between two items i and j; (b) The connection of two groups I and J, while $I = (i_1, i_2)$ and $J = (j_1, j_2)$.

Several tools have been developed for similarity calculation. For instance, DOSim [6], FunDO [7] are packages for calculating similarity between items in the field of bioinformatics. They are designed for particular projects, which makes them not easy to fit in other users' code without a deep understanding of their implementation. Moreover, many packages are designed for calculating similarity between two items [8,9]. In addition, most of tools for calculating similarity are published on R platform [10–13] or made as an online system [14]. There are little works have done with GIE model on Python platform.

Therefore, we present this independent, user-friendly Python package, which can calculate similarity between items and groups based on GIE model. Python is chosen as programming system because it is interpreted, object-oriented and open source, runs on all major operating systems and has become very popular for scientific programming [15]. GSimPy is based on the generic GIE model, so it can be easily integrated into other software as long as the data conforms to the GIE structure. GSimPy provides several methods to calculate similarity between items and groups, and it includes a number of methods to let users manage their customized data. Several visualization tools are also presented in this package to make the similarity results more readable.

2. GSimPy methods

According to the generic GIE model, item can be thought of as being annotated by a set of elements. Each item x has a set of elements. Let e represents element, $E(x)$ represents the elements set of item x, K represents the number of all elements. (x, e_i) is the relationship between item x and element e_i. M is a set which contains all the relationship mappings between items and elements. $E(x)$ can be defined as:

$$E(x) = \{e_i | (x, e_i) \in M, 1 \leq i \leq k\} \tag{1}$$

We apply Jaccard Index, also known as the Jaccard Similarity Coefficient, which is a statistic used for gauging the similarity and diversity of item sets. Items i and j both can be represented by a set of elements $E(i)$ and $E(j)$ respectively. By measuring similarity between two sets of elements of items, the similarity between items i and j can be obtained. We compute the similarity score of two items i and j as the Jaccard Index of $E(i)$ and $E(j)$:

$$Sim_{jaccard}(i, j) = \frac{|E(i) \cap E(j)|}{|E(i) \cup E(j)|} \tag{2}$$

Besides, Lin [16] thinks that the similarity between two sets $E(i)$ and $E(j)$ is determined by the common part and the different part between them. Similarity score is in proportion to the common part and inversely proportional to the different part [17]:

$$Sim_{Lin}(i, j) = \frac{2 \times |E(i) \cap E(j)|}{|E(i)| + |E(j)|} \tag{3}$$

From another perspective, item can be represented by a K-dimensional vector. Each dimension of vector represents an element, and the value of each dimension is the number of elements contained in the item. Items i and j can be represented by vector $I = (\alpha_1, \alpha_2, \ldots, \alpha_K)$ and $J = (\beta_1, \beta_2, \ldots, \beta_K)$ respectively. Cosine Similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of an angle between then. The Cosine similarity between items i and j can be formulated as follows:

$$Sim_{cosine}(i, j) = cos\theta = \frac{i \cdot j}{|i| \cdot |j|} = \frac{\sum_{x=1}^{K} \alpha_x \cdot \beta_x}{\sqrt{\sum_{x=1}^{K} (\alpha_x)^2} \times \sqrt{\sum_{x=1}^{K} (\beta_x)^2}} \tag{4}$$

Given two groups I and J defined as:

$$I = \{i_i, i_2, \ldots, i_M\} \tag{5}$$

$$J = \{j_i, j_2, \ldots, j_N\} \tag{6}$$

There are M and N items in group I and J respectively. The element sets of groups I and J are $E(I)$ and $E(J)$:

$$E(I) = \bigcup_{x=1}^{M} E(i_x) \tag{7}$$

$$E(J) = \bigcup_{y=1}^{N} E(j_y) \tag{8}$$

$Sim_{i,j}$ is the similarity of two items i and j, and it may be calculated with any of the similarity measures mentioned above. Similarity matrix S can be obtained, which contains all pairwise similarity scores of mappings item i of group I and mappings item j of group J.

$$S = [Sim_{i,j}]_{M \times N} \tag{9}$$

Three methods below consider the average, maximum and minimum values of the similarity matrix respectively to calculate group similarity.

Average method:

$$S_{average}(I,J) = \frac{1}{M \times N} \sum_{i \in I, j \in J} Sim_{i,j} \tag{10}$$

Complete method:

$$S_{complete}(I,J) = \max_{i \in I, j \in J} \left\{ Sim_{i,j} \right\} \tag{11}$$

Single method:

$$S_{single}(I,J) = \min_{i \in I, j \in J} \left\{ Sim_{i,j} \right\} \tag{12}$$

The rows and the columns of S represent two different directional comparisons, row vectors correspond to a comparison of I to J and column vectors of J to I. *rowScore* and *columnScore* as the averages over the row maxima and the column maxima of the similarity matrix S:

$$rowScore = \frac{1}{M} \sum_{i=1}^{M} \max_{j \in J} Sim_{i,j} \tag{13}$$

$$columnScore = \frac{1}{N} \sum_{j=1}^{N} \max_{i \in I} Sim_{i,j} \tag{14}$$

Using these definitions above, the *funSimMax* and *funSimAvg* methods consider the arithmetic maxima and average between *rowScore* and *columnScore* in similarity matrix S respectively [18], defined as:

$$S_{funSimMax}(I,J) = \max \{rowScore, columnScore\} \tag{15}$$

$$S_{funSimAvg}(I,J) = 0.5 \times (rowScore + columnScore) \tag{16}$$

Best-match average (BMA) method [19] considers the contributions from the similar items that from the two groups respectively. It can be represented as follows:

$$S_{BMA}(I,J) = \frac{\sum_{i=1}^{M} \max_{j \in J} Sim_{i,j} + \sum_{j=1}^{N} \max_{i \in I} Sim_{i,j}}{M+N} \tag{17}$$

Besides, when we only consider the relationship between group and elements, Eqs. (2), (3), (4) also can be used for calculating similarity between groups:

$$S_{jaccard}(I,J) = \frac{|E(I) \cap E(J)|}{|E(I) \cup E(J)|} \tag{18}$$

$$S_{Lin}(I,J) = \frac{2 \times |E(I) \cap E(J)|}{|E(I)| + |E(J)|} \tag{19}$$

$$S_{cosine}(I,J) = cos\theta = \frac{I \cdot J}{|I| \cdot |J|} = \frac{\sum_{x=1}^{K} \alpha_x \cdot \beta_x}{\sqrt{\sum_{x=1}^{K} (\alpha_x)^2} \times \sqrt{\sum_{x=1}^{K} (\beta_x)^2}} \tag{20}$$

3. Software description

GSimPy is implemented in Python 3.7. Dependencies of GSimPy include the packages NumPy 1.16.4, Matplotlib 3.1.1 and pyecharts 1.3.1; both very popular and stable open source Python packages. GSimPy consists of 3 sub-packages: data, sim and function (see Table 1).

3.1. Data sub-package

The data sub-package of GSimPy provides a number of methods for users to manage their customized data in database. Two tables *'item'* and *'group'* store item–element associations and group–item–element associations, respectively. The *'item'* table contains Item_ID field, Element_List field and Element_Num field, while the *'group'* table contains Group_ID field, Item_List field, Elememt_List field, Item_Num field and Element_Num field. Item–element data and group–item–element data can be easily stored in database by inputting *'item.txt'* document or *'group.txt'* document and using functions *write_item_into_database()* and *write_group_into_database()*. Fig. 2 instantiates the example of writing group data and item data from *'group.txt'* file and *'item.txt'* file to database *'sample.db'*. Each line is an association between item and element in *'item.txt'* file, while each line is an association between group and item in *'group.txt'* file. In addition, the data sub-package includes methods for adding, deleting, updating data in database (*insert_data()*,*delete_data()*,*update_data()*).

3.2. Sim sub-package

Several algorithms for calculating similarity between items and groups have been implemented in sim sub-package. It includes 3 algorithms for calculating item similarity (*The Jaccard Index*, *The Cosine Similarity*, *Lin's method*) and 9 algorithms for calculating group similarity (*average*, *single*, *complete*, *Jaccard*, *Cosine*, *Lin*, *funSimMax*, *funSimAvg*, *BMA*). These functions can support paired or grouped data input, return similarity score or similarity matrix respectively.

3.3. Function sub-package

This sub-package represents some useful functions. First, it gives a series of methods to query information from database. For instance, query information of items or groups (*get_group_info()*,*get_item_info()*), query the shared elements between two items (*get_2item_shared_element()*), query shared elements between two groups (*get_2group_shared_elements()*), query the number of all elements, items or groups (*get_all_element_num()*).

Moreover, we provide a function to build a GIE random model. In this GIE random model, the elements number of each item are unchanged, but its elements are randomly selected from all elements. Users can use the GIE random data generated by the GIE random model as a reference to evaluate similarity results. In addition, we implement 2 visualization methods—relation graph and heatmap. Examples of these two visualization methods are given in Fig. 3. Fig. 3(a) is a relation graph, where squares, triangles and circles represent groups, items and elements, respectively. In relation graph, the degree of an item is the number of elements associated with that item, while the degree of an element is the number of items annotated with that element. Heatmap can be used to easily compare similarity scores between items and groups. An example of it is shown in Fig. 3(b). The right side of the heatmap is the color band, which represents the mapping of values to colors. The left side of the heatmap is the similarity matrix, which uses color changes to reflect data information in a two-dimensional matrix. The darker the color in heatmap matrix, the higher the similarity between the two groups or items.

4. Illustrative examples

GSimPy is hosted at https://github.com/curlyae1995/GSimPy, but the default version can also be installed from PyPi using pip with the command:

python3 -m pip install GSimPy

GSimPy implementation is focused on providing an efficient interface between users and data. Because of this, it stands out for its simplicity and its ease for use. For example, querying information in database requires to know how to use SQL. But GSimPy manages all SQL-related inputs. Users only need to invoke related functions and input necessary parameters.

Table 1
A description of the core modules of GSimPy.

Sub-package name	Module name	Description
data	write_data_into_database.py	Write item or group data into database
	edit_data.py	Provide methods to add/delete/update data in database
sim	cal_itemsim.py	Contain 3 methods for calculating similarity between items
	cal_groupsim.py	Contain 9 methods for calculating similarity between groups
function	get_datainfo.py	Provide methods for querying database
	visualization.py	Two visualization tools to make the results more readable
	create_random_group.py	Generate random groups as a reference for similarity results

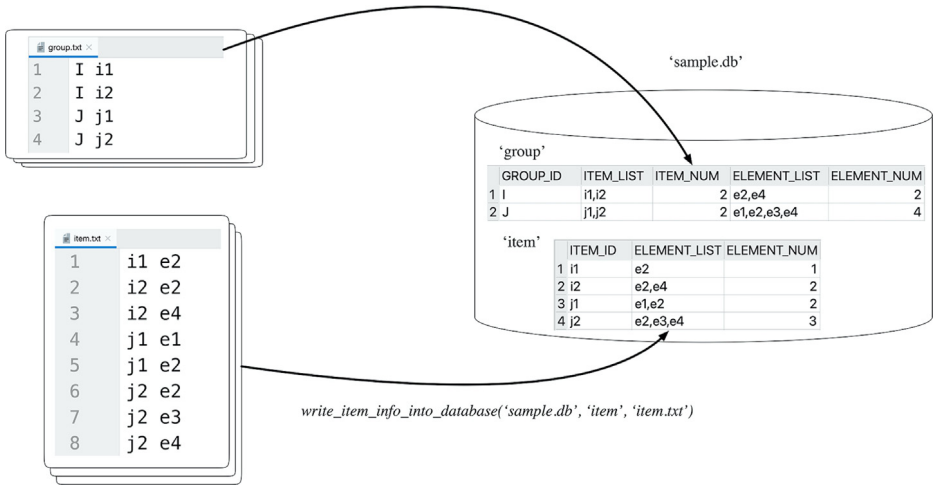
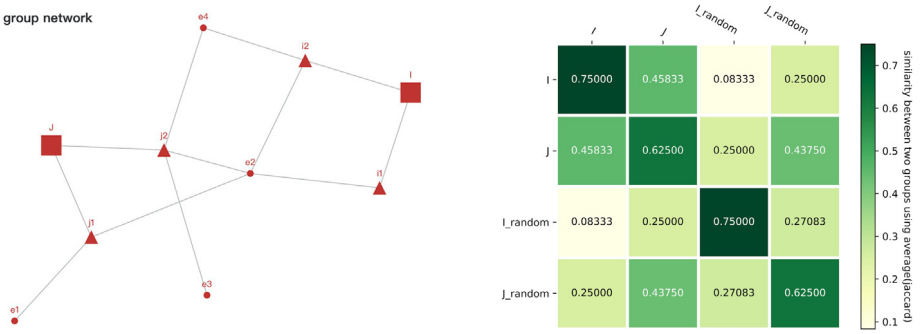


Fig. 2. Example of writing group and item data from *'group.txt'* and *'item.txt'* to database *'sample.db'*. Three parameters of *write_item_info_into_database()* are database name,item table name and the path of *'item.txt'* file. Three parameters of *write_group_data_into_database()* are database name, group table name and the path of *'group.txt'* file.



(a) (b)

Fig. 3. Two graphs produced by GSimPy. (a) Group network between groups *I* and *J*; (b) Heatmap of similarity matrix between groups *I*, *J*, *I_random* and *J_random*.

Take groups *I* and *J* in Fig. 1(b) as an example. The GIE data of Fig. 1(b) is stored in formatted plain text files *'group.txt'* and *'item.txt'*. First, we store the data in the database *'sample.db'*:

```
from data.write_data_into_database import *
write_item_info_into_database('sample.db', 'item', 'item.txt')
write_group_info_into_database('sample.db', 'group', 'group.txt')
```

Then all the 3 methods for calculating similarity between items and 9 methods for groups can be used to calculate group similarity between groups *I* and *J*. Choose The Jaccard Index for item similarity and average method for group similarity. The similarities between two items i_1 and i_2 and two groups *I* and *J* can be calculated.

```
from sim import cal_groupsim, cal_itemsim
sim_items = cal_itemsim.cal_2itemjaccard('sample', 'item', 'i1', 'i2')
print(sim_items)
⋙ 0.50000
sim_group = cal_groupsim.cal_2groupaverage('sample', 'group', 'I', 'J','jaccard')
print(sim_groups)
⋙ 0.4583
```

Two random groups *I_random* and *J_random* can be generated as a reference to evaluate similarity results of groups *I* and *J*. The elements number of each items from groups *I* and *J* are unchanged, but its elements are randomly selected from e_1, e_2, e_3 and e_4.

```
from function.create_random_reference import *
create_random_group('sample.db', 'group', 'item', 'I')
⋙ i1 ['e3'] i2 ['e4', 'e3']
⋙ ('I', ['i1', 'i2'], 2, ['e3', 'e4'], 2)
```

In the random group *I_random*, two items i_1 and i_2 are annotated by (e_3) and (e_3, e_4). *I_random* is composed of 2 items i_1 and i_2, and has 2 elements e_3 and e_4.

```
create_random_group('sample.db', 'group', 'item', 'J')
⋙ j1 ['e3', 'e1'] j2 ['e4', 'e2', 'e1']
⋙ ('J', ['j1', 'j2'], 2, ['e3', 'e1', 'e4', 'e2'], 4)
```

In the random group *J_random*, two items j_1 and j_2 are annotated by (e_1, e_3) and (e_1, e_2, e_4). *J_random* is composed of 2 items j_1 and j_2, and has 4 elements e_1, e_2, e_3 and e_4.

Store data into database *'sample.db'* as *I_random*, *J_random*.

```
from data import edit_data
edit_data.insert_data('sample.db', 'item')
edit_data.insert_data('sample.db', 'group')
```

GSimPy also provides users ability to visualize the relationship among element, item and group. Fig. 3(a) draw a relation graph to depicts the association between groups *I* and *J*. In Fig. 3(b), a heatmap is implemented for visualizing significant group associations.

```
from function.visualization import *
draw_relation_graph_between_groups('sample.db','group',
'I','J').render
draw_heatmap('sample.db','group','average','jaccard','I,J,
I_random,J_random','I,J,I_random,J_random')
```

Here, we take GOSemSim [12] as a comparison package of GSimPy. GOSemSim is an R package for semantic similarity [5] computation among Gene Ontology (GO) terms and sets of GO terms. It uses 'DO.db' as the data input file. Unlike the GIE module we defined, there is no element concept in this work. Think of GO terms as items and sets of GO terms as groups. This package uses goSim() based on semantic similarity and mgoSim() based on BMA method to calculate similarity between items and groups, respectively.

An example of using GOSemSim to calculate similarity between items and groups as follows:

```
> library(DO.db)
> library(GOSemSim)
> goSim("GO : 0004022", "GO : 0005515", ont = "MF",
measure = "Wang")
[1]0.252
> go_group1 = c("GO : 0004022", "GO : 0004024", "GO :
0004174")
> go_group2 = c("GO : 0009055", "GO : 0005515")
> mgoSim(go_group1, go_group2, ont = "MF", measure =
"Wang")
[1]0.299
```

5. Impact

GSimPy is the first publicly available Python package for calculating similarity between items and groups based on GIE model. Because GIE model is a highly abstract data structure, GSimPy can be applied to different research fields. For instance, in the field of biomedical, element, item and group in GIE model can be used to represent gene, disease and disease group, respectively. Using GSimPy can help to analyze relationships between diseases and disease groups.

The main impact of GSimPy is that it implements a complete set of methods for measuring group similarity based on GIE model and can be easily integrated into other Python projects. Having many different methods under one roof makes it possible to compare the similarity between items or groups using multiple similarity algorithms, which has been difficult before.

From the user perspective, the complex tasks of data management, similarity calculation, similarity results evaluation and data visualization will be simplified due to access to several methods to choose from, all contained in a single package.

6. Conclusions

In this paper, we present GSimPy, an open source and user-friendly Python package for measuring group similarity based on GIE model. It facilitates users to investigate items or groups relationships in many research fields. The functionality of the package as well as the source code is well documented. In the future, we plan to introduce more generic group models and similarity algorithms to enrich GSimPy, and try to develop related Python packages to use group similarity as a indicator in order to evaluate cluster algorithms such as K-means.

CRediT authorship contribution statement

Yifei Zhang: Software. **Jia Cao:** Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This project was supported by the National Natural Science Foundation of China(Grant No. 61602042).

References

[1] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surv 1999;31(3):264–323.
[2] Chen X, WU XS WW, et al. An improved initial cluster centers selection algorithm for K-means based on features correlative degree. J Sichuan Univ Eng Sci Ed 2015;47(1):13–9.
[3] Sun K, Gonçalves JP, Larminie C, Pržulj N. Predicting disease associations via biological network analysis. BMC Bioinform 2014;15(1):304.
[4] Bodenreider O, Aubry M, Burgun A. Non-lexical approaches to identifying associative relations in the gene ontology. In: Pacific symposium on biocomputing. Pacific symposium on biocomputing. NIH Public Access; 2005, p. 91–102.
[5] Pesquita C, Faria D, Falcao AO, Lord P, Couto FM. Semantic similarity in biomedical ontologies. PLoS Comput Biol 2009;5(7).
[6] Li J, Gong B, Chen X, Liu T, Wu C, Zhang F, et al. DOSim: an R package for similarity between diseases based on disease ontology. BMC Bioinform 2011;12(1):266.
[7] Osborne JD, Flatow J, Holko M, Lin SM, Kibbe WA, Zhu LJ, et al. Annotating the human genome with Disease Ontology. BMC Genomics 2009;10(1):S6.
[8] Mazandu GK, Mulder NJ. DaGO-Fun: tool for gene ontology-based functional analysis using term information content measures. BMC Bioinform 2013;14(1):284.
[9] Romero-Zaliz RC, Rubio-Escudero C, Cobb JP, Herrera F, Cordón Ó, Zwir I. A multiobjective evolutionary conceptual clustering methodology for gene annotation within structural databases: a case of study on the gene ontology database. IEEE Trans Evol Comput 2008;12(6):679–701.
[10] Yu G, Wang L-G, Yan G-R, He Q-Y. DOSE: an R/Bioconductor package for disease ontology semantic and enrichment analysis. Bioinformatics 2014;31(4):608–9.
[11] Zhou J, Shui Y, Peng S, Li X, Mamitsuka H, Zhu S. MeSHSim: An R/Bioconductor package for measuring semantic similarity over MeSH headings and MEDLINE documents. J Bioinform Comput Biol 2015;13(06):1542002.
[12] Yu G, Li F, Qin Y, Bo X, Wu Y, Wang S. GOSemSim: an R package for measuring semantic similarity among GO terms and gene products. Bioinformatics 2010;26(7):976–8.
[13] Yu G, Wang L-G, Han Y, He Q-Y. ClusterProfiler: an R package for comparing biological themes among gene clusters. Omics: J Integr Biol 2012;16(5):284–7.
[14] Hu Y, Zhao L, Liu Z, Ju H, Shi H, Xu P, et al. DisSetSim: an online system for calculating similarity between disease sets. In: 2016 IEEE international conference on bioinformatics and biomedicine. IEEE; 2016, p. 1641–52.
[15] Van Rossum G, Drake FL. The python language reference manual. Network Theory Ltd.; 2011.
[16] Lin D, et al. An information-theoretic definition of similarity. In: Icml, vol. 98. 1998. p. 296–304.

[17] Su S, Zhang L, Liu J. An effective method to measure disease similarity using gene and phenotype associations. Front Genet 2019;10.
[18] Schlicker A, Domingues FS, Rahnenführer J, Lengauer T. A new measure for functional similarity of gene products based on Gene Ontology. BMC Bioinform 2006;7(1):302.
[19] Pesquita C, Faria D, Bastos H, Ferreira AE, Falcão AO, Couto FM. Metrics for GO based protein semantic similarity: a systematic evaluation. BMC Bioinform 2008;9(5):S4.