Package ‘Diderot’

November 6, 2017

Type Package
Title Bibliographic Network Analysis
Version 0.12
Date 2017-10-30
Author Christian Vincenot
Maintainer Christian Vincenot <christian@vincenot.biz>
Depends R (>= 3.0.1), foreach, graphics, igraph
Imports stringi, RCurl, splitstackshape, data.table, utils, stats, parallel, doParallel
Description Enables the user to build a citation network/graph from bibliographic data and, based on modularity and heterocitation metrics, assess the degree of awareness/cross-fertilization between two corpora/communities. This toolset is optimized for Scopus data.
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2017-11-06 06:17:15 UTC

R topics documented:

Diderot-package .................................................. 2
build_graph .................................................... 5
compute_BC_ranking ........................................... 6
compute_citation_ranking ...................................... 8
compute_custom_modularity ................................... 9
compute_Ji ..................................................... 10
compute_Ji_ranking ........................................... 11
compute_modularity .......................................... 12
create_bibliography ........................................ 13
get_date_from_doi .......................................... 15
get_reference_title ......................................... 16
heterocitation ................................................ 17


**Description**

*Denis Diderot* (1713-1784), French philosopher and co-founder of the modern encyclopedia.

This package allows to detect and quantify the unification or separation of two bibliographic corpora through the creation of citation networks. This tool can be used to study the spread of concepts across scientific disciplines, or the fusion/fission of scientific communities.

**Details**

<table>
<thead>
<tr>
<th>Package:</th>
<th>Diderot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
</tbody>
</table>
A typical flow of use of the package includes the following points.

First, literature metadata, including references, from the two fields of studies to analyze are downloaded from Scopus (or built manually). This data is imported to create a bibliographic dataset using `create_bibliography`.

Second, a graph is created with a call to `build_graph` to reproduce the citation network in the bibliographic dataset.

Finally, statistical analysis can be performed on the graph to assess the fusion/fission state of the two corpora/communities. Heterocitation indices (i.e. share and balance) show how much publications or authors cite papers from the other corpus (see `heterocitation` and `heterocitation_authors` respectively). Such analysis shall always be preceded by a call to `precompute_heterocitation` to perform initial calculations. These metrics are completed by traditional as well as custom modularity metrics (see `compute_modularity` and `compute_custom_modularity` respectively) that translate how much the communities are separated. Publications that foster mutual awareness and cross-fertilization between the corpora/communities can be identified using the usual betweenness centrality metric (see `compute_BC_ranking`) and the Ji index (see `compute_Ji_ranking`).

Author(s)

Christian Vincenot

Maintainer: Christian Vincenot (christian@vincenot.biz)

See Also

`igraph`

Examples

```r
## Not run:
# Two corpora on individual-based modelling (IBM) and agent-based modelling (ABM)
# were downloaded from Scopus. The structure of each corpus is as follows:
tt<-read.csv("IBMmerged.csv", stringsAsFactors=FALSE)
str(tt,strict.width="cut")
```
# publications (keys).
labels<-c("IBM","ABM")
keys<-c("individual-based model\individual based model",
"agent-based model\agent based model")

# Build the IBM-ABM bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c("IBMmerged.csv","ABMmerged.csv"),
labels=labels, keywords=keys)

### [1] "File IBMmerged.csv contains 3184 records"
### [1] "File ABMmerged.csv contains 9641 records"

# Build and save citation graph
gs<-build_graph(db=ds,small.year.mismatch=T,fine.check.nb.authors=2, attrx=c("Corpus","Year","Authors","DOI"))

### [1] "Graph built! Execution time: 1200.22 seconds."
save_graph(gs, "graph.graphml")

# Compute and plot modularity
compute_modularity(gs, 1987, 2018)

### [1] 0.3164805
plot_modularity_timeseries(gs, 1987, 2018, window=1000)

# Compute and plot publication heterocitation
gr_sx<-precompute_heterocitation(gs, labels=labels, infLimitYear=1987, supLimitYear=2018)

### [1] "-----------------------------------------------"
### [1] "IBM     ABM     IBM|ABM"
### [1] "1928     5378    153"

### [1] "Edges summary"

### [1] "-----------"
### [1] "IBM->IBM/IBM->Other 5583/1086 => Prop 0.163"
### [1] "ABM->ABM/ABM->Other 16946/2665 => Prop 0.136"
### [1] "General Same/Diff 22529/3751 => Prop 0.143"

### [1] "Heterocitation metrics"

### [1] "--------------------------"
### [1] "Sx ALL / IBM / ABM"
### [1] "0.127 / 0.137 / 0.124"
### [1] "Dx ALL / IBM / ABM"
### [1] "-0.652 / -0.803 / -0.598"

heterocitation(gs, labels=labels, 1987, 2005)

### [1] "Sx ALL / ABM / IBM"
### [1] "0.047 / 0.214 / 0.007"
### [1] "Dx ALL / ABM / IBM"
### [1] "-0.927 / -0.690 / -0.982"

plot_heterocitation_timeseries(gs, labels=labels, mini=-1, maxi=-1, cesure=2005)

# Compute author heterocitation
het<-heterocitation_authors(gs, 1987, 2018, pub_threshold=4)
head(het[order(het$avgDx,decreasing=T),c(1)], n=10)

### [6] "Reinhardt J.W." "Ng K." "Kazanci C." "Senior A.M." "Ariel G."
build_graph

Builds a citation graph.

Description

Builds a citation graph based on a database of bibliographic records generated with create_bibliography. This process is automatically parallelized on multicore hardware. By default, matching between title and references is done based on the full title, publication year, and three first authors. Publication attributes present in the dataframe can be copied to graph nodes using the attrs argument.

Usage

build_graph(db, title = "Cite Me As", year = "Year", authors = "Authors", ref = "Cited References", set.title.as.name = F, attrs = NULL, verbose = F, makeCluster.type = "PSOCK", nb.cores=NA, fine.check.threshold = 1000, fine.check.nb.authors = 3, small.year.mismatch = T, debug = F)

Arguments

db Bibliographic database created with create_bibliography.
title Name of the data frame column in which publication titles are listed.
year Name of the data frame column in which publication years are listed.
authors Name of the data frame column in which publication authors are listed.
ref Name of the data frame column in which publication references are listed.
set.title.as.name Set graph vertex ID to publication title
attrs Attributes of the bibliographic database (i.e. data frame column names, such as "Authors", "Year") to be set as vertex attributes.
verbose Verbosity flag triggering a more detailed output during graph building.
makeCluster.type Type of cluster to be used to parallelize the graph building process. For more options, see makeCluster in the doParallel library.
nb.cores Number of cores to be used for parallel computation.
compute_BC_ranking

Function to compute the Betweenness Centrality ranking.

Description

This function computes the Betweenness Centrality (BC) for each graph node (i.e. publication).

fine.check.threshold
Title length under which citation matching is further confirmed based on publication year. This value can be reduced to increase performance on large bibliographic databases. By default, publication year check is always performed.

fine.check.nb.authors
Maximum number of authors to check against for citation matching. This value can be reduced to increase performance on large bibliographic databases. Default value is three authors.

small.year.mismatch
Flag indicating whether small year mismatches (+/- 1 year) should be tolerated. It is recommended to keep this flag to TRUE to accommodate usual inconsistencies in bibliographic databases.

debug
Debug flag allowing the user to browse function calls upon execution error. For more details, see recover in the utils library.

Value
Returns a graph object.

Author(s)
Christian Vincenot (christian@vincenot.biz)

See Also
create_bibliography

Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db, small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)
compute_BC_ranking

Usage

compute_BC_ranking(gr, labels, write_to_graph = F)

Arguments

gr      Citation graph
labels  Labels (i.e. names) of the two corpora featured in the graph.
write_to_graph  Flag to indicate whether to write results to the graph (i.e. save BC values as node attributes).

Value

If write_to_graph is FALSE, returns a list of entries (authors, title, year, corpus, BC) sorted by decreasing BC. Else, returns the graph given as input to which BC values are added as node attributes.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

compute_citation_ranking, compute_Ji_ranking

Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
                           labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attr=c("Corpus","Year","Authors"), nb.cores=1)

# Compute BC
compute_BC_ranking(gr, labels)
Function to compute the citation ranking.

**Description**

This function computes the citation count for each graph node (i.e. publication).

**Usage**

```r
compute_citation_ranking(gr, labels, write_to_graph = F)
```

**Arguments**

- `gr`: Citation graph
- `labels`: Labels (i.e. names) of the two corpora featured in the graph.
- `write_to_graph`: Flag to indicate whether to write results to the graph (i.e. save citation count values as node attributes).

**Value**

If `write_to_graph` is `FALSE`, returns a list of entries (authors, title, year, corpus, citations) sorted by decreasing citation count. Else, returns the graph given as input to which citation count values are added as node attributes.

**Author(s)**

Christian Vincenot (christian@vincenot.biz)

**See Also**

- `compute_BC_ranking`
- `compute_Ji_ranking`

**Examples**

```r
labels<-c("Corpus1","Corpus2")

db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
                         labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

# Compute Citation Ranking
compute_citation_ranking(gr, labels)
```
compute_custom_modularity

Function to compute the custom modularity of a citation graph over a time window.

Description

This function computes custom modularity of the citation graph. Custom modularity here stands for Newman’s modularity computed over the subgraph comprising nodes that belong to a single corpus only and are within the time window, as well as direct outgoing neighbors of the former nodes (whatever their year tag). Basically, the citation graph considered thus includes publications within the time window as well as older papers that they cite.

Usage

compute_custom_modularity(gr, infLimitYear, supLimitYear)

Arguments

- gr: Citation graph
- infLimitYear: Start year of the time window considered (included)
- supLimitYear: End year of the time window considered (*excluded*)

Value

Returns the custom modularity value of the subgraph restricted to the interval [infLimitYear;supLimitYear[.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

compute_modularity, plot_modularity_timeseries

Examples

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
                          labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db,small.year.mismatch=TRUE, attrc=c("Corpus","Year","Authors"), nb.cores=1)
```
compute_Ji

Function to compute the evolution of the Ji metric for a term (e.g. publication title) present in the reference list of a bibliographic dataset.

Description

This function computes the Ji metric for a given term from a bibliographic dataset and returns its annual evolution within the timeframe specified. This metric indicates how much the term (e.g. publication title, software name) is cited simultaneously in the references of both corpora and is thus important for cross-fertilization between the two communities. This function is run on the bibliographic dataset (created with `create_bibliography`) and is thus useful before graph creation or when the term to be searched is not the title of a node in the resulting graph. For instance, if the user knows that a publication (or, e.g. software or scientific database referenced only through a URL or grey literature) is cited and may have an impact on cross-fertilization between the two communities (the literature of which is represented by the two corpora) but does not have its own entry in the bibliographic database and would therefore not be featured as a node in the graph created by `build_graph`, the `compute_Ji` function can be used to assess its importance.

Usage

```r
compute_Ji(db, pubtitle, labels, from = -1, to = -1)
```

Arguments

- `db`: Bibliographic database created with `create_bibliography`.
- `pubtitle`: Publication title, or more generally term to be searched (e.g. software name).
- `labels`: Labels (i.e. names) of the two corpora featured in the graph.
- `from`: Start year of the time window considered (included)
- `to`: End year of the time window considered (*excluded*)

Value

Dataframe containing year and Ji metric value.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

`create_bibliography`
compute_Ji_ranking

Examples

```
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
                             labels=labels, keywords=NA)

# Compute Ji
compute_Ji(db, "Title1", labels, from=1990, to=2018)
```

compute_Ji_ranking  Function to compute the Ji metric ranking for publications in a citation graph.

Description

This function computes the Ji metric for each graph node (i.e. publication). This metric indicates how much a publication is cited simultaneously by both corpora and is thus important for cross-fertilization between the two communities.

Usage

```
compute_Ji_ranking(gr, labels, inflimitYear, suplimitYear, write_to_graph=F)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gr</td>
<td>Citation graph</td>
</tr>
<tr>
<td>labels</td>
<td>Labels (i.e. names) of the two corpora featured in the graph.</td>
</tr>
<tr>
<td>inflimitYear</td>
<td>Start year of the time window considered (included)</td>
</tr>
<tr>
<td>suplimitYear</td>
<td>End year of the time window considered (<em>excluded</em>)</td>
</tr>
<tr>
<td>write_to_graph</td>
<td>Flag to indicate whether to write results to the graph (i.e. save Ji values as node attributes).</td>
</tr>
</tbody>
</table>

Value

If write_to_graph is FALSE, returns a list of entries (authors, title, year, corpus, citations from corpus 1, citation from corpus 2, Ji) sorted by decreasing Ji. Else, returns the graph given as input to which Ji are added as node attributes.

Author(s)

Christian Vincenot (christian@vincenot.biz)
compute_modularity

See Also

`build_graph`, `precompute_heterocitation`, `compute_ji`

Examples

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2), labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

# Compute Ji ranking
compute_ji_ranking(gr, labels, 1990, 2018)
```

### Description

This function computes Newman’s modularity of the citation graph restricted to a given time window and ignoring nodes belonging to both corpora simultaneously.

#### Usage

```r
compute_modularity(gr, infLimitYear, supLimitYear)
```

#### Arguments

- `gr` : Citation graph
- `infLimitYear` : Start year of the time window considered (included)
- `supLimitYear` : End year of the time window considered (*excluded*)

#### Value

Returns Newman’s modularity value of the subgraph restricted to the interval `[infLimitYear;supLimitYear]`.

#### Author(s)

Christian Vincenot (christian@vincenot.biz)

#### See Also

`compute_custom_modularity`, `plot_modularity_timeseries`
Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempf1,tempf12),
                           labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db, small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

# Compute Modularity
compute_modularity(gr, 1990, 2018)

create_bibliography Function to create a bibliographic dataset

Description

This function creates a bibliographic dataset based on two external corpus files, each representing the bibliography of a given domain.

Usage

create_bibliography(corpora_files, labels, keywords, retrieve_pubdates = F,
                     clean_refs = F, encoding = NULL)

Arguments

corpora_files Vector containing the paths to two corpus files (e.g. Scopus exports). The CSV files should contain for each record at least Authors (comma separated), Publication Title, Publication Year, and References (semicolon separated). The inclusion of DOI (for date checking; see the retrieve_pubdates option) as well as Abstract, Author.Keywords, and Index.Keywords (for the in-depth identification of publications belonging to both corpora) are strongly recommended.

labels Labels (i.e. names) given to the two corpora to be analyzed.

keywords Keywords identifying the two corpora

retrieve_pubdates Flag indicating whether to confirm publication dates by retrieving them (see get_date_from_doi)

clean_refs Attempt to clean references and keep titles only. NOT RECOMMENDED, especially if build_graph should be used subsequently.

encoding Character encoding encoding used in the input files.
**Value**

Returns a dataframe containing a bibliographic dataset usable by Diderot and including all references from both corpora.

**Author(s)**

Christian Vincenot (christian@vincenot.biz)

**See Also**

build_graph

**Examples**

```r
## Not run:
# Two corpora on individual-based modelling (IBM) and agent-based modelling (ABM)
# were downloaded from Scopus. The structure of each corpus is as follows:
tt<-read.csv("IBMmerged.csv", stringsAsFactors=FALSE)
str(tt,strict.width="cut")
### 'data.frame': 3184 obs. of 9 variables:
### $ Authors : chr "Chen J., Marathe A., Marathe M." "Van Dijk D., Sl1"
### $ Title : chr "Coevolution of epidemics, social networks, and in"
### $ Year : int 2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ... 
### $ DOI : chr "10.1007/978-3-642-12079-4_28" "10.1016/j.procs.20"
### $ Link : chr "http://www.scopus.com/inward/record.url?eid=2-s2.."
### $ Abstract : chr "This research shows how a limited supply of antiv"
### $ Author.Keywords: chr "Antiviral; Behavioral economics; Epidemic; Microe"
### $ Index.Keywords : chr "Antiviral; Behavioral economics; Epidemic; Microe"
### $ References : chr "(2009) Centre Approves Restricted Retail Sale of ".

# Define the name of corpora (labels) and specific keywords to identify relevant
# publications (keys).
labels<-c("IBM","ABM")
keys<-c("individual-based model|individual based model",
        "agent-based model|agent based model")

# Build the IBM-ABM bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c("IBMmerged.csv","ABMmerged.csv"),
  labels=labels, keywords=keys)
### [1] "File IBMmerged.csv contains 3184 records"
### [1] "File ABMmerged.csv contains 9641 records"

# Processed output. Note the field name changes (for standardization with ISI Web
# of Knowledge format) and addition of the "Corpus" field (with identification of
# joint "IBM | ABM" publications based on keywords).
str(db, strict.width="cut")
### 'data.frame': 12504 obs. of 10 variables:
### $ Authors : chr "Chen J., Marathe A., Marathe M." "Van Dijk D., Sloot "
### $ Cite Me As : chr "Coevolution of epidemics, social networks, and indivi"
### $ Year : int 2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ... 
### $ DOI : chr "10.1007/978-3-642-12079-4_28" "10.1016/j.procs.2010.0"
```
**get_date_from_doi**

Function to retrieve publication date based on Digital Object Identifier (DOI)

**Description**

This function retrieves precise publication date by querying the Digital Object Identifier (DOI) web server. Alternatively, if `extract_date_from_doi` is set to TRUE, the function will first try to extract a publication year from the publication DOI string. If `create_bibliography` is called with `retrieve_pubdates = TRUE`, it calls `get_date_from_doi` for each record to confirm publication dates.

**Usage**

```r
get_date_from_doi(doi, extract_date_from_doi)
```

**Arguments**

- **doi**: Character string representing the Digital Object Identifier (DOI) of the publication
- **extract_date_from_doi**: Flag indicating whether to try to simply extract publication year from the DOI string before resorting to online queries to the DOI server

**Value**

Returns a date in YYYY-MM-DD format or YYYY-MM format if `extract_date_from_doi` is set to TRUE.

**Note**

Scopus records already contain the year of publication of scientific papers indexed. However, in some cases these are inaccurate and can be verified by comparing them with the date retrieved by this function. Note that

**Author(s)**

Christian Vincenot (christian@vincenot.biz)
get_reference_title

See Also

create_bibliography

Examples

```r
## Not run:
# Query publication date from DOI server
get_date_from_doi(doi="10.1016/j.procs.2010.04.250",extract_date_from_doi=TRUE)

## End(Not run)
# Extract date from DOI string
get_date_from_doi(doi="10.1016/j.procs.2010.04.250",extract_date_from_doi=TRUE)
```

get_reference_title  Function to extract the publication title from a reference using Freecite

Description

Function to extract the publication title from a reference using an online query to Freecite

Usage

```r
get_reference_title(str)
```

Arguments

| str            | Character string representing a reference |

Value

Returns a character string of the publication title

Author(s)

Christian Vincenot (christian@vincenot.biz)
Description

This function calculates the heterocitation share and heterocitation balance between two corpora A and B in the time window specified. The heterocitation share (Sx) of a publication belonging to corpus A is defined as the percentage of citations to publications belonging to corpus B (or A|B) in its reference list. The global heterocitation share for corpus A is calculated as the average heterocitation share of the publications that corpus A contains (e.g. a value of 0.2 for corpus A indicates that, on average, publications in corpus A cite only 20% of papers from corpus B). The heterocitation balance metric (Dx), on the other hand, takes into consideration the respective sizes of corpus A and B to discern how much the heterocitation share deviates from values expected in the case of well-mixedness (i.e. if A and B originated from a unique community; e.g. a value of -50% for corpus A indicates that, on average, publications in corpus A cite papers from corpus B half less frequently than expected, which suggests a lack of mutual awareness between the corpora and related communities).

Usage

heterocitation(gr, labels, inflimitYear, suplimitYear)

Arguments

- gr: Citation graph priorly preprocessed with `precompute_heterocitation`
- labels: Labels (i.e. names) of the two corpora featured in the graph.
- inflimitYear: Start year of the time window considered (included)
- suplimitYear: End year of the time window considered (*excluded*)

Value

Returns a numerical vector containing, in this order, the heterocitation share (Sx) for corpus A, B and global, and the heterocitation balance (Dx) for A, B and global.

Note

`precompute_heterocitation` should be called before running this function.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

`precompute_heterocitation, plot_heterocitation_timeseries, heterocitation_authors, MC_baseline_distribution, significance_Dx`
**Examples**

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
    labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=("Corpus","Year","Authors"), nb.cores=1)

# Heterocitation
gr<-precompute_heterocitation(gr,labels, 1990, 2018)
heterocitation(gr,labels, 1990, 2018)
```

---

**heterocitation_authors**

*This function computes heterocitation metrics for authors*

---

**Description**

This function computes heterocitation metrics for authors. The heterocitation share ($S_x$) and heterocitation balance ($D_x$) of an author are calculated as the average of these metrics for papers published by this author within the given time window. See the man page of `heterocitation` for definitions of heterocitation metrics.

**Usage**

```r
heterocitation_authors(gr, inflimitYear, suplimitYear, pub_threshold = 0, remove_orphans = F, remove_citations_to_joint_papers = F)
```

**Arguments**

- `gr`  
  Citation graph priorly preprocessed with `precompute_heterocitation`
- `inflimitYear`  
  Start year of the time window considered (included)
- `suplimitYear`  
  End year of the time window considered (*excluded*)
- `pub_threshold`  
  Minimum number of publications for authors to be considered.
- `remove_orphans`  
  Do not consider publications that do not cite any other paper in the dataset (i.e. orphan nodes in the citation network)
- `remove_citations_to_joint_papers`  
  Do not consider publications belonging to both corpora in the authors’ average corpus calculation.
load_graph

Function to load a citation graph

Description

This function loads a citation graph saved on the filesystem.

Value

Returns a data frame containing author name ("Authors"), number of publications ("NbPubs"), list of publication years ("Years"), list of publications corpora ("Corpus"), list of publication heterocitation share ("Sx"), list of publication heterocitation balance ("Dx"), average heterocitation share ("avgSx"), average heterocitation balance ("avgDx"), average corpus value of publications ("avg-Corpus"), regression coefficient of the heterocitation share evolution ("coeffSx"), regression coefficient of the heterocitation balance evolution ("coeffDx"), regression coefficient of the evolution of the corpus value of publications ("coeffCorpus").

Note

precompute_heterocitation should be called before running this function.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

precompute_heterocitation, heterocitation

Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfi1,tempfi2),
labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

# Heterocitation
gr<-precompute_heterocitation(gr,labels, 1990, 2018)

# Author heterocitation
heterocitation_authors(gr, 1990, 2018)
Usage

load_graph(filename)

Arguments

filename File to load

Value

Returns a graph object.

Note

This function basically supports only graph previously saved with Diderot’s save_graph. However, as the file is actually a graphml file handled by igraph, advanced users may use this function on appropriate graphs created elsewhere, as long as they respect Diderot’s structure (presence of a "Corpus" field, etc).

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

save_graph

Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfi1,tempfi2),
                            labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

## Not run:
save_graph(gr, "Saved.graphml")

# Load saved graph
gr<-load_graph("Saved.graphml")

## End(Not run)
**MC_baseline_distribution**

*Function to compute baseline heterocitation values for the graph under study with random permutation of corpus attributions*

---

**Description**

This function performs Monte Carlo runs with random permutations of corpus tags in the graph provided and computes the heterocitation balance on the new graphs. Permutation is repeated over several iterations (set through the "rep" argument) and provides a baseline Dx values for the graph topology considered. This can then be compared with the Dx value obtained for the original graph to evaluate whether it could merely be the result of chance (see `significance_Dx`).

**Usage**

```r
MC_baseline_distribution(gr, labels, infYearLimit, supYearLimit, rep = 20)
```

**Arguments**

- `gr` : Graph file (created with `build_graph`)
- `labels` : List of the names of the two corpora studied (e.g. `c("Computer Science", "Mathematics")`), present in the "Corpus" attribute
- `infYearLimit` : Minimum year considered in this study
- `supYearLimit` : Maximum year considered in this study
- `rep` : Number of Monte Carlo iterations

**Value**

This function currently plots the histograms of distribution of Dx values generated through random permutations of corpus tags among the records. Returns a list containing:

- `Dx1` : Dx value for corpus 1 per iteration
- `Dx2` : Dx value for corpus 2 per iteration
- `DxALL` : Global Dx value per iteration

**Author(s)**

Christian Vincenot (christian@vincenot.biz)

**See Also**

`significance_Dx, heterocitation`
nb.refs  

*Function to calculate the number of citations in a bibliographic database*

**Description**

This function outputs the number of records and citations in a bibliographic database, and returns the latter.

**Usage**

```r
nb.refs(db)
```

**Arguments**

- `db` : Bibliographic database

**Value**

Returns the number of citations in the bibliographic database.

**Author(s)**

Christian Vincenot (christian@vincenot.biz)

**Examples**

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfi1,tempfi2),
                          labels=labels, keywords=NA)

# NB refs
nb.refs(db)
```
plot_authors_count

Function to plot authors count timeseries

Description

This function plots and returns the annual number of new authors and cumulative number of authors in the bibliographic database.

Usage

plot_authors_count(db)

Arguments

db Bibliographic database created with created_bibliography.

Value

Returns a dataframe containing year, annual number of new authors (i.e. not seen before), and cumulative number of authors.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also
compute_citation_ranking

Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
                           labels=labels, keywords=NA)

# Plot authors count
## Not run: plot_authors_count(db)
plot_heterocitation_timeseries

Function to plot the heterocitation share and heterocitation balance timeseries

Description

This function plots and returns annual heterocitation share and heterocitation balance values.

Usage

plot_heterocitation_timeseries(gr_arg, labels, mini = -1, maxi = -1, cesure = -1)

Arguments

- **gr_arg**: Citation graph
- **labels**: Labels (i.e. names) of the two corpora featured in the graph.
- **mini**: Start year of the time window
- **maxi**: End year of the time window
- **cesure**: Year before which values should be cumulated. Default value is -1, which indicates that each year in the time window should be plotted.

Value

Returns a dataframe with year and annual values for heterocitation share (sx1, sx2 and sxall for corpus A and B and global resp.) and heterocitation balance (dx1, dx2 and dxall for corpus A and B and global resp.).

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

precompute_heterocitation, heterocitation

Examples

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempf1,tempf2),
                         labels=labels, keywords=NA)
```
plot_modularity_timeseries

Function to plot graph modularity timeseries

Description

This function plots and returns annual graph modularity values for predefined corpora (representing communities). See compute_modularity for details on modularity calculation.

Usage

plot_modularity_timeseries(gr_arg, mini = -1, maxi = -1, cesure = -1, window = 1, modularity_function = "normal")

Arguments

- **gr_arg**: Citation graph
- **mini**: Start year of the time window
- **maxi**: End year of the time window
- **cesure**: Year before which values should be cumulated. Default value is -1, which indicates that each year in the time window should be plotted.
- **window**: The temporal sliding window size over which modularity should be computed.
- **modularity_function**: Modularity function to be used for the calculation: "custom" indicates that compute_custom_modularity will be used, whereas "normal" indicates that compute_modularity will be used.

Value

Returns a dataframe containing year and annual modularity value.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

compute_modularity, compute_custom_modularity
Examples

```
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
                         labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db, small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

# Compute Modularity timeseries
plot_modularity_timeseries(gr, 1990, 2018)
```

Description

This function plots and returns the annual number of publications.

Usage

```
plot_publication_curve(gr, labels, k = 1)
```

Arguments

- **gr**: Citation graph
- **labels**: Labels (i.e. names) of the two corpora featured in the graph.
- **k**: Text font size (multiplier of cex values)

Value

Returns a dataframe containing year and annual publication count for each corpus and both together.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

- `compute_citation_ranking`
**precompute_heterocitation**

*This function precomputes heterocitation values for each node/publication of a graph.*

### Description

This function computes heterocitation values for each publication and stores them as node attributes in the graph. The heterocitation share of a publication belonging to corpus A is defined as the percentage of citations to publications belonging to corpus B (or A|B) in its reference list (e.g. a value of 0.2 for a publication in corpus A indicates that the publication cites only 20% of papers from corpus B). The heterocitation balance metric, on the other hand, takes into consideration the respective sizes of corpus A and B to discern how much the heterocitation share deviates from values expected in the case of well-mixedness (i.e. if A and B originated from a unique community; e.g. a value of -30% for a publication in corpus A indicates that it cites papers from corpus B 30% less frequently than expected).

### Usage

```r
precompute_heterocitation(gr, labels, infLimitYear, supLimitYear)
```

### Arguments

- **gr**: Citation graph
- **labels**: Labels (i.e. names) of the two corpora featured in the graph.
- **infLimitYear**: Start year of the time window considered (included)
- **supLimitYear**: End year of the time window considered (*excluded*)

### Examples

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempf1,tempf12),
    labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

# Publication curve
plot_publication_curve(gr,labels)
```
Value

Returns the graph gr with added node attributes Sx and Dx representing the heterocitation share and heterocitation balance respectively.

Note

Corpus-wide heterocitation values can be computed using heterocitation.

Author(s)

Christian Vincenot (christian@vincenot.biz)

See Also

heterocitation, plot_heterocitation_timeseries, compute_Ji_ranking

Examples

```r
labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempfil1,tempfil2),
labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)
gr<-precompute_heterocitation(gr,labels, 1990, 2018)
```

save_graph  

Function to save a graph

Description

This function saves a graph produced with Diderot. The resulting structure is actually a graphml file and can thus be exported to third party software.

Usage

```r
save_graph(gr, filename)
```

Arguments

- **gr**: Graph object to save.
- **filename**: File to save the graph to.
significance_Dx

Author(s)
Christian Vincenot (christian@vincenot.biz)

See Also
load_graph

Examples

labels<-c("Corpus1","Corpus2")

# Build a bibliographical dataset from Scopus exports
db<-create_bibliography(corpora_files=c(tempf1,tempf2),
                           labels=labels, keywords=NA)

# Build graph
gr<-build_graph(db=db,small.year.mismatch=TRUE, attrs=c("Corpus","Year","Authors"), nb.cores=1)

## Not run: save_graph(gr, "Saved.graphml")

significance_Dx Function to evaluate the significance of the heterocitation balance value

Description
This function assesses to what extent the heterocitation balance (Dx value) calculated for a graph departs from baseline situation. The latter typically represents Dx values to be expected by chance, i.e. through random permutation of corpus assignation at the node/vertex level (see MC_baseline_distribution). A Shapiro-Wilk test is first executed on the control distribution (using shapiro.test) and if the normality hypothesis is not rejected, a one-sample t test (see t.test) is used to test whether value is significantly different from the control distribution. The strength of this difference is additionally assessed through Glass’ delta, an estimator of effect size (Glass, McGraw, and Smith, 1981).

Usage
significance_Dx(value, control, normality_threshold=0.05)

Arguments
value Heterocitation balance (Dx) calculated for the citation network studied
control Baseline distribution of Dx values in control experiments
normality_threshold P value threshold under which the hypothesis of normality is rejected in the preliminary Shapiro-Wilk test
Value

Returns a list containing the p-value obtained in a one-sample t test comparing value and the control distribution (with null hypothesis being that value could come from the control distribution) or NA if the control distribution is not normal based on a Shapiro-Wilk normality test, and Glass’ estimator of effect size.

Author(s)

Christian Vincenot (christian@vincenot.biz)

References


See Also

significance_Dx, heterocitation

Examples

## Not run:
# Heterocitation in our graph
heterocitation(gr_sx, labels=labels, 1987, 2005)
### [1] "Sx ALL / ABM / IBM"
### [1] "0.047 / 0.214 / 0.007"
### [1] "Dx ALL / ABM / IBM"
### [1] "-0.927 / -0.698 / -0.982"

# Generate a baseline distribution for Dx values obtained through chance
# Here, we run 200 iterations of node corpus permutations
baseline<-MC baseline_distribution(gr_sx, labels, 1987, 2018, 200)

# Assess whether our observed Dx is possibly due to chance
significance_Dx(-0.927, baseline[["Dx ALL"]])
### [1] "Distribution is normal. Performing t-test."
###
### One Sample t-test
###
### data:  value - control
### t = -3.23, df = 319, p-value < 2.2e-16
### alternative hypothesis: true mean is not equal to 0
### 95 percent confidence interval:
### -0.9159834 -0.9048923
### sample estimates:
### mean of x
### -0.904379
###
### [1] "Glass' effect size: -18.0563442219448"
## End (Not run)
Index

*Topic \textasciitilde kwd1
  build\_graph, 5
  compute\_BC\_ranking, 6
  compute\_citation\_ranking, 8
  compute\_custom\_modularity, 9
  compute\_ji, 10
  compute\_Ji\_ranking, 11
  compute\_modularity, 12
  create\_bibliography, 13
  get\_date\_from\_doi, 15
  get\_reference\_title, 16
  heterocitation, 17
  heterocitation\_authors, 18
  load\_graph, 19
  MC\_baseline\_distribution, 21
  nb\_refs, 22
  plot\_authors\_count, 23
  plot\_heterocitation\_timeseries, 24
  plot\_modularity\_timeseries, 25
  plot\_publication\_curve, 26
  precompute\_heterocitation, 27
  save\_graph, 28
  significance\_Dx, 29

*Topic \textasciitilde kwd2
  build\_graph, 5
  compute\_BC\_ranking, 6
  compute\_citation\_ranking, 8
  compute\_custom\_modularity, 9
  compute\_ji, 10
  compute\_Ji\_ranking, 11
  compute\_modularity, 12
  create\_bibliography, 13
  get\_date\_from\_doi, 15
  get\_reference\_title, 16
  heterocitation, 17
  heterocitation\_authors, 18
  load\_graph, 19
  MC\_baseline\_distribution, 21
  nb\_refs, 22

Author Hetero\_citation
  (heterocitation\_authors), 18

Author Heterocitation
  (heterocitation\_authors), 18

Betweenness Centrality
  (compute\_BC\_ranking), 6

Bibliographic Network Creation
  (create\_bibliography), 13
  build\_graph, 5, 10, 12\textendash14

Citation Graph Building (build\_graph), 5

Citation Graph Creation
  (create\_bibliography), 13

Citation Graph Loading (load\_graph), 19

Citation Ranking
  (compute\_citation\_ranking), 8
  compute\_BC\_ranking, 3, 6, 8
  compute\_citation\_ranking, 7, 8, 23, 26
  compute\_custom\_modularity, 3, 9, 12, 25
  compute\_ji, 10, 10, 12
  compute\_Ji\_ranking, 3, 7, 8, 11, 28
  compute\_modularity, 3, 9, 12, 25
  create\_bibliography, 3, 6, 10, 13, 15, 16

Custom Modularity
  (compute\_custom\_modularity), 9

Diderot (Diderot\_package), 2

Diderot\_package, 2

get\_date\_from\_doi, 13, 15, 15
get_reference_title, 16
Graph Saving (save_graph), 28

Hetero-Citation (heterocitation), 17
Hetero-citation Timeseries
  (plot_heterocitation_timeseries), 24
heterocitation, 3, 17, 18, 19, 21, 24, 28, 30
Heterocitation Significance
  (significance_Dx), 29
Heterocitation Timeseries
  (plot_heterocitation_timeseries), 24
heterocitation_authors, 3, 17, 18

igraph, 3

Ji Metric (compute_Ji), 10
Ji Ranking (compute_Ji_ranking), 11

load_graph, 19, 29

makeCluster, 5
MC_baseline_distribution, 17, 21, 29
modularity (compute_modularity), 12
Modularity Timeseries
  (plot_modularity_timeseries), 25

nb_refs, 22

plot_authors_count, 23
plot_heterocitation_timeseries, 17, 24, 28
plot_modularity_timeseries, 9, 12, 25
plot_publication_curve, 26
precompute_heterocitation, 3, 12, 17–19, 24, 27
Publication Timeseries
  (plot_publication_curve), 26

recover, 6

save_graph, 20, 28
shapiro.test, 29
significance_Dx, 17, 21, 29, 30

t.test, 29