USER GUIDE

SDG publication labelling using Romero et al. thesaurus

Step 0 – Prepare working location
Define a folder where you will work (the working directory). Inside this folder you have to create a folder called ‘Data’ in which the publication data, downloaded from Web of Science in .txt format, is located. Open R and set the working directory to the folder where you will work. You can load the functions into R by using the line “load(‘Commands.RData’)”. Make sure that the following packages are installed: “parallel”, “Matrix”, “igraph”, “textstem”, “plyr”, “dplyr”, “ggplot2” and “tm”.

Step 1 – Data preparation
First, for analysing information in WoS we start from the files (in plain text) as exported by WoS. In this first step a matrix is created with the information related to each paper, where each publication is defined by its “UT” (accession number), a unique identifier.

If your data is already in matrix form, you can skip this step. However, make sure “##” is used as split character between bibliographic items (BI). Usually, WoS uses a semicolon to separate individual BI, but sometimes some BI may have semicolons as part of their structure (e.g. Author.Year.Journal,Serie;volume) which would cause conflicts. If you omit this step, make sure that your matrix is located in a folder named ‘Analysis’ in your working directory.

In this step you will use the function ReadFiles() to read and organise the plain text files. The function requires as input the desired name for the output file. Notice that you must *not* indicate the extension of the file (this will be .csv by default). For example, you can use ReadFiles(‘output_name’), but not ReadFiles(‘output_name.csv’). After executing the function, the output file can be found in a folder named ‘Analysis’, inside your working direction, that is generated by the function in this step.

In your R environment execute the function “ReadFiles(‘output_name’)”

Step 2 – Co-bibliography network
Once the data matrix is created, the next step is to create the co-bibliography network. For this purpose we will use the function “CoBibMatrix()”, which requires two inputs. First, you must include the name of the matrix which contains the data, i.e., the matrix we created in the first step (e.g., ‘output_name’). Notice that the extension is not included in the input. The second input in this function is the number er of processing cores that you use. Given that many of the calculations included in this guide are computationally intensive, the code is designed to use parallel computing. In this part, for each core you will require 1-2 Gb or RAM memory depending on the size of your dataset. If you are working in a Unix-based OS (e.g., MAC-OS, Linux) you can use as many cores as available in your computer, otherwise (i.e., Windows) you can only use 1 core. After executing this function you will find a “*_network_full.graphml” file that contains the co-bibliography network including all the links possible.

In your R environment execute the function “CoBibMatrix(‘matrix_name’, ‘cores’)

Step 3 – Analysing the co-bibliography network links
Not all the links in the co-bibliography network are meaningful, which means you need to define a threshold value that indicates after which value you will consider a link as meaningful. The function “Threshold.Eval()” is used to evaluate different thresholds. This

function requires five input values. The first input is the network file to be analysed, i.e., the network created in step 2. The second input is the number of cores you will use. For Unix-based OS users it is recommended to use half or a quarter of the cores that are available. For Windows use 1 core. The last three input values indicate the sequence of values to evaluate as threshold. Input three is the initial value, input four the final value. Input five is the step size in the sequence. For instance, if you use “Threshold.Eval(‘network_name’, ‘cores’, 0, 100, 0.1)” the analysis will output the results for 1100 threshold values between 0 and 100. It is recommended to use “Threshold.Eval(‘network_name’, ‘cores’, 10, 25, 0.5)” for experimentation and “Threshold.Eval(‘network_name’, ‘cores’, 5, 30, 0.1)” for the final analysis.

The output of the function is a file named “*_T-Eval.csv” in which you find how the network properties vary as a function of the different threshold values. Typically, the threshold is identified using a balance between the maximum number of communities (column ‘#Com’), the maximum modularity (column ‘Modularity’) and the number of nodes (column ‘Nodes’). Frequently, the value for the threshold is around 17-19. The columns of the output file are defined as follows: ‘Nodes’ are the number of nodes in the cleaned graph, ‘#Com’ is the number of communities (based on the Louvain clustering algorithm) in the cleaned graph, ‘Modularity’ is the modularity score of the communities in the cleaned graph, ‘#Com.10’ is the number of communities that have more than 10 publications, ‘NodesCom.10’ is the number of publications in the graph if all communities with less than 10 publications are removed, ‘Triangles’ are the number of triangles in the cleaned graph.

In your R environment execute the function “Threshold_Eval(‘network_name’, ‘cores’, ‘min_val’, ‘max_val’, ‘step’)”

Step 4 – Cleaning the data
After establishing the threshold value for the meaningful links, the data needs to be cleaned (i.e., removing non-meaningful links and the related publications). In this step the communities are added to the data as well, based on the Louvain clustering algorithm.

The function “Threshold_Cleaner()” cleans the data. It requires three inputs, first the name of the network file to be cleaned, i.e., the network created in step 2. Second, the threshold value (based on the threshold evaluation in step 3, e.g., 17.8) and third, the name of the output file for the cleaned data (e.g., ‘clean’). You can use the same name for input and output, but it is not recommended as the data will be overwritten in that case. As a result, you will find a .csv (matrix file with the publication data) and .graphml (network file) file with the name of output.

In your R environment execute the function “Threshold_Cleaner(‘network_name’, ‘threshold_value’, ‘output_name’)”

Step 5 – Keyword search
In this step we will search the publication data for the keywords related to the SDGs, as defined in the thesaurus developed by Romero et al. (2021). The function searches for the keywords in the title, abstract and keywords (TI, AB, DE) of the publications. This step can be concatenated with step 1 or step 4, depending on whether all publications should be searched or only publications in the cleaned network.

The function “Search_function()” is used to search for the keywords in the publications. It requires three inputs. First, the name of the cleaned dataset (in matrix form, which has the extension .csv, e.g., ‘clean’ from step 4), then the name of the CSV file with the thesaurus (this file must be in the ‘Analysis’ folder), and finally the number of cores you will use. You can use as many as you have available, except for Windows (use 1). The function output is a dataset with the number of keywords per SDG for each publication and the most probable

SDG assigned to each publication, up to a maximum of three SDGs (for details see the Main Manuscript-Method). Notice that only the publications in which keywords are found are in this dataset. Furthermore, the function gives as output a file named '*-Words_Searched' which identifies which word was mapped in which publication.

In your R environment execute the function "Search_Function('matrix_name', 'thesaurus_name', 'cores')"

Step 6 – SDG triads in the network
In this step the SDG triads in the network are determined, i.e., the relations between the different SDGs and their categories according to the transformative lens framework by Schot et al. (2018). The function "SDG_Triads()" determines the SDG triads in the network and the corresponding categories (Sociotechnical Systems (ST), Transversal Directionalities (TD), Framework Conditions (FC)).

The function requires four input values. First the database with the SDG publications from step 5. The second input is the network file which contains at least all the SDG publications from the database (i.e., the full network from step 2, when for step 5 the full database is used, or the cleaned network from step 4, when for step 5 the cleaned database is used). The third input value is the output name, and the fourth input value is the number of cores.

The output of the function are two datasets, one with the SDG triads in the whole network and one with the SDG triads in each community. Notice that if in step 5 the full database from step 1 is used, the network from step 2 does not contain communities and the file with the triads per community will not be created.

In your R environment execute the function "SDG_Triads('input_file', 'input_network', 'output', 'cores')"

Step 7 – Define SDG communities
In this step the SDG communities are defined. The requirements for a community to be labelled as SDG community depends on the share of SDG publications, both over all the years in the analysis and the share of SDG publications in the last quarter of the years in the analysis (e.g., for an analysis from 2000-2020 the share of SDG publications from 2015-2020).

The function "SDG_Communities()" requires six inputs. First, the database with the labelled SDG publications (as created in step 5). Second, the cleaned database which includes all publications and communities in the network (as created in step 4). Third and fourth are the cut-off points for the SDG publication share in a community (for details see the Main Manuscript-Method). For an initial analysis these cut-off points can be set to 0 and the barplots will be plotted, allowing for a visual analysis to determine the cut-off points. The fifth and sixth one are for the barplots, whether these should be saved (default is 'yes') and if yes, the name for the figure (default is 'sdg_share').

The function creates the file "*_sdg_com.csv" for both input files, where the column 'sdg_com' is added, marking the SDG communities. The function also creates a folder 'Images' in the working directory and if 'save_fig' is 'yes' it will be placed in this folder.

In your R environment execute the function "SDG_Communities('file_sdg', 'file_data', 'T4_Cut', 'Total_Cut', 'save_fig, 'figname')"

Step 8 – HCPC clustering
To determine the high-level clusters of both the SDG and the non-SDG communities and their top 3 of most frequent research areas, according to WoS.

Make sure to install the package “FactoMineR” before executing the function. The function HCPC_Clustering() takes three input values. First the database which includes the SDG communities, i.e., the database from step 7. Second and third are the number of SDG and non-SDG clusters that should be created, which is by default set to six clusters. Typically, this number is around 6-7 clusters.

The function updates the input dataset with four columns, ‘Cluster’ – which includes a ‘Y’ if it is a high-level SDG cluster and a ‘N’ if not, and the number of the cluster, and ‘SC1-3’ – three columns including the three main research areas of the cluster (where SC1 is the most prominent one).

In your R environment execute the function “HCPC_Clustering(‘input_file’, ‘Clust_SDG’, ‘Clust_noSDG’)”