Brain, Digital, & Learning

pISSN : 2384-2474
eISSN : 2586-7490

RESEARCH ARTICLE

Keyword-based Scoring for The Prioritization of Differentially Expressed Genes Using Gene Ontology Annotations and Network Propagation

Songhwan Kim, Dongsun Park*
Laboratory of Animal Physiology and Medicine
Department of Biology Education, Korea National University of Education

차등 발현 유전자 데이터 분석을 위한 유전자 온톨로지와 네트워크 전파를 활용한 키워드 기반 유전자 스코어링 방식

김성환, 박동선*
한국교원대학교 생물교육과, 동물생리의학실험실

*Corresponding Author: Dongsun Park, dvmdpark@knue.ac.kr

ABSTRACT

Computational gene prioritization provides a basis for the utilization of high-throughput expression data; however, methods for prioritization considering relevance to both biological processes and selected keywords are lacking. In this study, a method was developed to rank differentially expressed genes (DEGs) by utilizing the Gene Ontology (GO) database for keyword searches and propagating the results through a protein–protein interaction network. A scoring system that effectively biases the scores of genes relevant to given keywords for avoidance and preference was established. This scoring method was combined with scoring based on expression characteristic groups (ECGs) with network propagation to obtain a final combined score (cScore). The performance of the new approach was evaluated using a rat middle cerebral artery occlusion (MCAO) dataset, revealing that the method more effectively filtered out DEGs compared with conventional methods based on both significance and fold change values, excluding 76% of genes in average, while retaining genes of interest. Further improvements, including addressing the inability of downward accumulation to balance terms with conflicting hits and the limited number of databases utilized for scoring, can further improve the performance of the method. Overall, the newly developed method can improve the interpretation of DEG data.

Key words: Gene prioritization, network propagation, gene ontology, protein–protein interaction network, keyword search

Brain, Digital, & Learning
2023, Vol. 13, No. 4, 415-451
https://doi.org/10.31216/BDL.20230026

Received: November 14, 2023
Revised: December 16, 2023
Accepted: December 17, 2023

Introduction

The emergence of next-generation sequencing platforms has not only allowed us to explore the genomes of vast and diverse organisms, but has also made it possible to design experiments focusing on changes in the quantity of genetic materials (Chen et al., 2011; Day-Williams & Zeggini, 2011; Mortazavi et al., 2008), including analyses of differentially expressed genes (DEGs) (Oshlack et al., 2010). However, vast data resulting from sequencing are difficult to interpret (Pervez et al., 2022). It is common to use computational methods to process and analyze such data (Stark et al., 2019).

Several enrichment methods and databases have been developed for assigning genes to categories or groups based on meaningful biological information (Khatri & Drăghici, 2005). Such enrichment-based approaches are very effective for obtaining an overview of either the state of organisms or differences in biological functions in experimental data (e.g., DEGs) (Hinderer et al., 2019).

To gather information on the differences between samples at the transcriptome level, DEG identification is a crucial step (Young et al., 2012). DEGs provide a basis for discovering novel targets correlated with certain conditions or can be used for further enrichment (Chen et al., 2011). Although a number of enrichment methods are available, it is often necessary to inspect data for each gene individually, e.g., for genes that are unannotated or when the composition or quantity of data is not suitable for common computational methods (Khatri & Drăghici, 2005). In such cases, the log-fold change or P-value of statistical tests and its variants are used to prioritize (or rank) each gene before manual inspection (Patterson et al., 2006; Peart et al., 2005; Raouf et al. 2008). However, in a few cases, log-fold change values do not completely represent the relevance of a gene of interest or the results of statistical tests (Claverie, 1999; Baldi & Long, 2001).

In this study, we propose a scoring method that is keyword-based and targets the Gene Ontology (GO) (Ashburner et al., 2000; The Gene Ontology Consortium, 2023) annotation system utilizing a relational graph structure, performing a downward accumulation of scores. However, annotation and the linkage between genes and GO terms are not complete, meaning that the extent of certain ontological information will influence gene scoring (Khatri & Drăghici, 2005). To address this issue, we applied the scores to network propagation, a widely used gene-prioritization method (Raj & Sreeja, 2018), for further processing. Utilizing a protein–protein interaction network (PPI) (Szklarczyk et al., 2023) limits the applicability to transcriptome data but minimizes the effect of insufficient annotation and maximizes the scoring of relevant genes (Chen et al., 2009; Oti et al., 2006). The proposed method is applied to rat middle cerebral artery occlusion (MCAO) model datasets using code written in the R environment. To rank genes, scores are calculated based on ontological relevance to keywords for avoidance and search and merged with a score that utilizes only network propagation based on expression characteristics. Various scoring-ranking methods are compared to verify the effectiveness of the proposed method.

Materials and Methods

Animals

Eight-week-old male Sprague-Dawley rats (n = 10/group) weighing 300–330 g were purchased from Daehan Biolink (Eumseong, Korea). Rats were housed in an environmentally controlled room with a constant temperature (23 ± 3 °

C) and relative humidity (50 ± 10%), and a 12 h light/dark cycle. The rats were fed a standard rodent diet and purified water ad libitum. All animal experiments were performed in accordance with the standard operation procedures of the Laboratory Animal Center, Chungbuk National University (CBNU), Korea. The study protocol was approved by the Institutional Animal Care and Use Committee of the CBNU (#CBNUA-1527-21-02).

Middle Cerebral Artery Occlusion (MCAO) Model Preparation

Silicone-coated threads were prepared according to a previously described method (Schmid-Elsaesser et al., 1998). Briefly, polyethylene tubing (Intramedic, Batavia, IL, USA) with an internal diameter of 0.28 mm was filled with silicone. A 4/0 monofilament nylon suture (Ailee, Busan, Korea) was inserted 5 mm into one end of the polyethylene tubing (Choi et al., 2012). After insertion, the tube and encased silicone were cut using a razor blade at a point 0.5 mm beyond the tip of the nylon suture, and the silicone was dried for 24 h. Immediately prior to surgery, the polyethylene tube was removed from the nylon suture, leaving a uniform coating of silicone bonded to the distal 5 mm.

Focal cerebral ischemia was induced in rats as described previously (Longa et al., 1989) with slight modifications (Choi et al., 2012; Kim et al., 2020; Park et al., 2012). Briefly, rats were anesthetized with isoflurane and restrained in the supine position. A midline incision was made on the ventral side of the neck, exposing the left common carotid artery (CCA), external carotid artery (ECA), and internal carotid artery (ICA). The ICA and the lower part of the CCA were blocked using a clip, and the upper part of the ECA was ligated. Subsequently, an opening was made in the middle of the ECA, and a silicone-coated thread was introduced through the opening. The thread was advanced 18 mm through the ICA up to the origin of the MCA. After occlusion was achieved, the thread was secured using a ligature, and the incision was sutured.

Transcriptome Sequencing Analysis

RNA samples from the brain in each group, normal control (Nor), 0.5 h after MCAO, and 3 h after MCAO, were sent to Macrogen Inc. (Seoul, Korea) for transcriptome sequencing. The TruSeq Stranded mRNA LT Sample Prep Kit (Illumina, San Diego, CA, USA) was used to prepare a cDNA library according to the TruSeq Stranded mRNA Sample Preparation Guide (Part # 15031047 Rev. E). DNase was used to remove DNA contamination. The poly-A mRNAs were then purified and broken down into short fragments. Cleaved RNA fragments were used to synthesize cDNA using random hexamer primers. After purification, the cDNA fragments were ligated to universal adapters containing sequencing priming sites. Subsequently, the fragments were amplified using PCR and analyzed using agarose gel electrophoresis. Fragments with insert sizes between 200 bp and 400 bp were subjected to deep sequencing. For paired-end sequencing, both ends of the cDNA were sequenced using the Illumina NovaSeq 6000 instrument (Illumina). For quality control of the raw reads, the overall read quality, total bases, total reads, GC content (%), and basic statistics were calculated. To reduce bias in the analysis, artifacts such as low-quality reads, adaptor sequences, contaminant DNA, and PCR duplicates were removed using Trimmomatic (Bolger et al., 2014). The sliding window method was used, and the bases of reads that did not qualify within a window size of four and a mean quality of 15 were trimmed (Bolger et al., 2014). Reads shorter than 36 bp were removed to produce trimmed data. The trimmed reads were aligned using

HISAT2 version 2.1.0 (Johns Hopkins University, Baltimore, MD, USA). Reads mapped to each gene were counted using StringTie version 2.1.3b (Johns Hopkins University, Baltimore, MD, USA), which measures the abundance of each transcript in FPKM. Differentially expressed genes (DEGs) analysis was done by edgeR (Robinson et al., 2010) package in R for each group-wise comparison. P value was calculated by exact test and adjusted by Benjamini-Hochberg correction (Benjamini & Hochberg, 1995).

Scoring Workflow

All steps in the proposed method are described in detail in the following section. The proposed method can be divided into four major subprocesses: (1) gather all required external files from each database and reading the user input, (2) check for keyword hits and propagate downwards in the GO graph to obtain scores, (3) perform network propagation within the PPI network using the scores from the previous step, and (4) combine all scores and apply additional filtration defined by the user prior to analyses. An overview of the workflow is presented in Fig. 1.

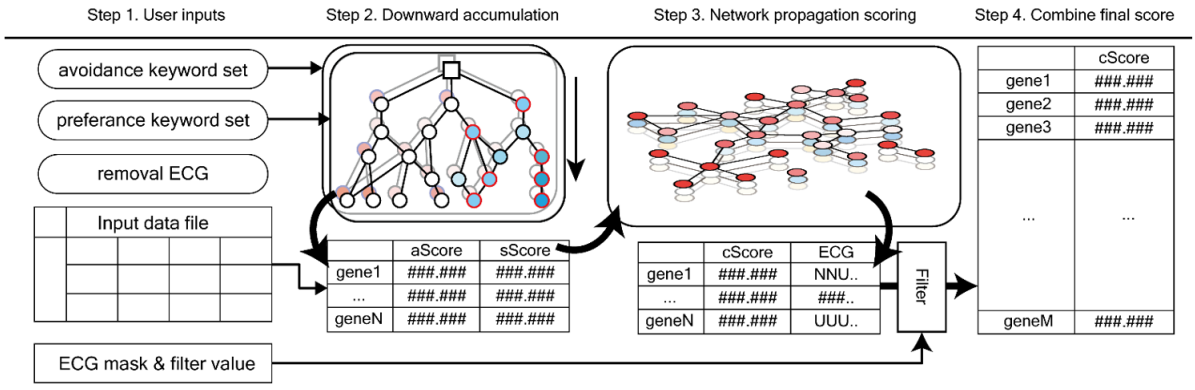


Fig. 1. Overview of the complete workflow of suggested methods.

User Inputs

To apply scoring to multiple datasets, the following input data are required: Gene ID in the form of the NCBI gene ID and one or more DEG datasets (fold-change data and significance values from statistical tests). Additional information can be included, such as fragments per kilobase of transcript per million mapped reads (FPKM), transcripts per million mapped reads (TPM), and each threshold for filtering. As transcript ID and gene ID do not always get mapped 1:1, it is advised to reduce the data such that there is no duplicate information in further processes.

The input keywords for avoidance and search share the same structure, where the user can freely apply multiple conditions using the logical operators AND (&), OR (|), and NOT (!) between keywords in quotation marks. The operations are grouped in parentheses to indicate their order. Additionally, overriding can be set when both avoidance and search hits are observed for a single term. Users can define the range of match-hit operations within the names, definitions, and synonyms of GO terms.

For each set of experiments, genes were assigned to one of three expression characteristic groups (ECGs): significantly upregulated (U), significantly downregulated (D), or non-significant expression change (N). Thus, for a given number

of experiments (n), the number of combined expression characteristic groups can reach 3n. Assignment to an expression characteristic group is based on the significance and fold-change thresholds input by the use and is used globally for the specific run. This creates a group system that is similar to the Venn diagram approach but allows for a more detailed approach to grouping each gene. Users can specify a set of characteristic expression groups that will be used as seeds for subsequent in-data gene network propagation. The r values are adjustable for each network propagation, as described in detail in [1-3. Network Propagation and Scoring].

For this example, keywords of avoidance was set to words related to inflammation and immune response such as "Inflamma", "Immun", "Interleukin", "Lymphocyte". Keywords of preference were set to words related to neurons such as "Neuro", "Synap", "Nervous". (See Appendix, user input section)

Search Keyword Hits & Downward Accumulation in GO Graph

Each string for avoidance and search, consisting of individual keywords and operations, was tokenized and stored separately. All keywords were matched individually to each GO term based on a defined range of fields and successful hits were recorded. Hit information was combined with the logical operation that was formerly tokenized and stored and finally outputted if a GO term matched the given condition. Subsequently, a downward accumulation of scores was performed. For the total structure of GO terms and annotations, we used following files, "GO-basics.obo", and "rgd.gaf.gz" which were retrieved from the Gene Ontology website (2023-10-09) and processed. For the reference gene list, data were retrieved from the Rat Genome Database (RGD [Rattus norvegicus (Norway rat) NCBI:txid10116, Genes] GENES.RAT.txt, Rat Genome Database Web Site, Medical College of Wisconsin, Milwaukee, Wisconsin. World Wide Web (URL: http://rgd.mcw.edu/) [Oct, 2023]). Regarding how GO terms are related to each other, forming an acyclic graph with a hierarchical order, any child term of a term that has a hit is relevant to the keyword set used. Thus, it is possible to accumulate the hit scores for the parent terms downward along the GO graph for child terms. For all GO terms with hits starting with a score of 1, each child term is assigned a score representing the accumulation of scores preceding its parent terms. Where SGO is a set of all GO term,; for any term T_i in SGO, a subset ch(T_i) and par(T_i) can be defined to as follows.

$$ch(T_i) = \{T \in S_{GO} | T\, being\, child\, term\, of\, T_i\}$$
$$par(T_i) = \{T \in S_{GO} | T\, being\, parent\, term\, of\, T_i\}$$

Setting S_{hit} as a subset of S_{GO} containing all terms with hits, the downward passing score for term T_i is as follows:

$$Score_{dp}(T_i) = \frac{n(ch(T_i) \cap S_{hit})}{n(ch(T_i))}$$

The final accumulated score for each term T_i in S_{GO} can be calculated as follows:

$$Score_a(T_i) = \sum_{T \in par(T_i)} Score_{dp}(T) + \begin{cases} 1 & if\ T_i \in S_{hit} \\ 0 & if\ T_i \notin S_{hit} \end{cases}$$

If only the terms with hits are scored, all scores for terms that fall in ($S_{GO} \cap S_{hit}^{C}$) will be masked to 0.

By propagating downwards in the GO graph and going beyond simple searches and hits, it is possible to obtain higher scores with more parent terms with hits (i.e., a higher probability of being relevant to the keyword set) or parent terms that are lower in the hierarchy in the GO graph (i.e., more specific). Enabling extended scoring, terms with no hits at the initial stage receive cumulative scores. Thus, it is possible to incorporate terms that are not directly relevant to the keyword set but have biological relevance.

For any gene *g* in the given dataset, the score for the term match-hit is the maximum of all annotated terms' $Score_a$s.

$$annot(g_i) = \{T \in S_{GO} | T\, is\, annotated\, to\, g_i\}$$
$$Score_g(g_i) = max(\{Score_a(T) : T \in annot(g_i)\})$$

All calculated $Score_g$s are then passed on to the next step for network propagation.

Network Propagation and Scoring

A PPI network was instantiated using STRING from the DB in the reference library. Each gene can be referred to as a "node" and one node is connected to another node with an "edge" if it has a higher confidence score than the threshold (where the default threshold is a combined score of >700) in the original PPI network. The completed set of nodes and edges is then converted into an adjacency matrix for subsequent use. Network propagation from the retrieved graph was performed using the R package "diffusr" and the Markov random walk (MRW) method was used.

Where p^t is the vector at the current time step t, p^0 is the initial value vector, A' is the stochastic matrix converted from the previous adjacency matrix by column normalization, and r being the restart rate, the vector of the next time step p^{t+1} can be calculated as follows:

$$p^{t+1} = (1 - r)A'p^t + rp^0$$

This process is repeated iteratively until the maximum number of iterations (by default set to 100,000) or when the L1 norm difference between p^{t+1} and p^t is less than the threshold (by default set to 10^{-10}). For term search and avoidance, the initial vector p^0 is initialized using L1 normalized set of $Score_g$ containing the annotation scores for all genes. In the case of ECGs avoidance, all nodes assigned to the avoidance group are assigned an initial value of 1 and the L1 normalized vector is passed to p^0. The values of p^{t+1} at the end of the random walk are recorded for each corresponding gene.

ECGs of removal assigned for this experiment were ("NNN","NND","NDN","NDD","DNN","DND","DDN","DDD"), aiming to exclude genes those where down-regulated or had no significant expression difference.

Merging Scores & Filtration

Because each score set produced by the random walk had a high probability of having different statistical characteristics owing to the different sizes of the actual data and the size of the library set for network propagation, L1 normalization for each score set was performed. Then, each score set was identified as either "preferred" (denoted as P) or "avoided" (denoted as A). The final combined score vector combined-score (cScore) can be described as follows:

$$cScore = \sum_{S_p \in P} \left(\frac{S_p}{n(P)} * W_p \right) - \sum_{S_a \in A} \left(\frac{S_a}{n(A)} * W_a \right)$$

W_i is a weight multiplier for each corresponding score set that can be defined by the user (default to 1).

The calculated cScore can be used to rank genes in the context of their relevance to input keywords and expression characteristics. Further filtration is possible using other criteria to improve interpretation.

No additional criteria were set for this experiment, and the filtration was solely done by cScore and ECGs where genes with cScore greater than or equal to 0, and ECG being in the list of ("UUU", "UNU") are kept.

Development Environment

Programming to implement the proposed method (see Appendix) was performed using R statistical software (v4.3.1; R Core Team, 2023), tidyverse (Wickham et al. 2019), fastmatch (Urbanek, 2023), diffusr (Dirmeier, 2018), STRINGdb (Szklarczyk et al., 2023), Go.db (Carlson, 2023), and GOfuncR (Grote, 2023).

Results

Comparison of Filtering Power with Conventional Methods

In Fig. 2, Number of filtered DEGs for each method is shown, the total background numbers of unique genes being 37,991. P value only, genes with corrected p values < 0.05 were considered significant. P values and fold changes combined, P values < 0.05 and |log fold change| > 1 were the thresholds for significance. (Combination of up-regulated and down-regulated incorporates log fold change over 1 and below -1 respectively.) cScore & ECG mask filtered based on cScore > 0 and selected by the predefined ECG for each dataset.

As shown in Fig. 2, the most DEGs were detected in the comparison between the Nor and 3 h samples, followed by the comparison between 0.5 h and 3 h. Based on the experimental design, these results showed that the majority of DEGs detected were related to the progression of brain damage. Using the combination of two datasets, the number of filtered DEGs in 0.5 h/3 h & Nor/3 h were larger those for the other comparisons, revealing that the majority of genes associated with brain damage have much higher expression differences at the late stage (0.5 h to 3 h). Accordingly, it would be logical to choose the Nor/0.5 h & Nor/3 h dataset and filter significantly upregulated genes for manual inspection to maximize possibility of detecting genes that are highly expressed at the early stage of brain damage.

Table 1 shows the percentage decrease in filtered DEGs, and it is apparent that the power of reduction using the cScore + ECG mask is the greatest in datasets incorporating only single data. This is because the conventional method is based on significance values and fold changes, which are more conservative in multiple combined datasets. Each additional data point resulted in a stricter condition.

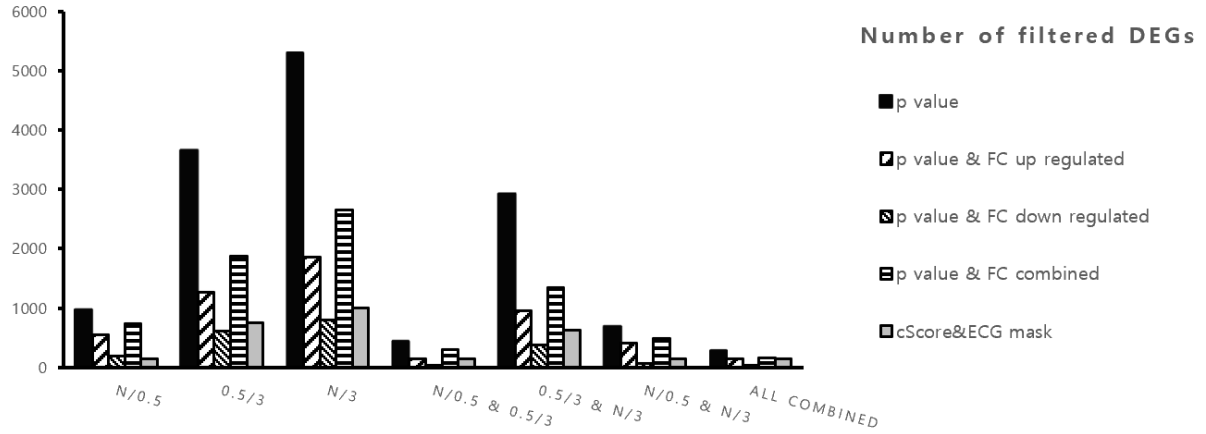


Fig. 2. Number of filtered DEGs from each method. X-axis represents composition of each dataset, such as N/0.5 being a dataset where p value, fold change (FC), ECG, and cScore were calculated on DEG analysis of sample "0.5" to sample "N" and filtered. "&" between two datasets indicate that genes were filtered matching conditions for both datasets. "ALL COMBINED" dataset is filtered by combining all "N/0.5", "0.5/3", and "N/3".
(Note: N = Normal sample, 0.5 = 0.5 hour after MCAO sample, 3 = 3 hours after MCAO sample)

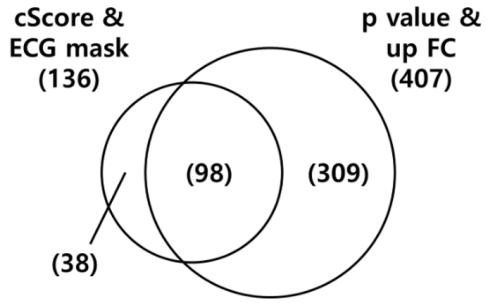
Table 1. Percentage decreased filtered DEGs of (cScore + ECG mask) method compared to (p value only) method and (p value and fold-change combined) method. Each name of datasets indicates the composition of each dataset, such as N/0.5 being a dataset where p Val, FC, ECG and cScore were calculated on DEG analysis of sample "0.5" to sample "N". Combination of datasets with "&" indicates that filter conditions were matched for both datasets. "ALL COMBINED" being dataset filtered by combining all "N/0.5", "0.5/3", and "N/3" dataset.
(Note: p Val = P value, FC = fold-change , N = Normal sample, 0.5 = 0.5 hour after MCAO sample, 3 = 3 hours after MCAO sample)

Dataset	cScore + ECG mask / p Val			cScore + ECG mask / p Val + FC		
	Individual Case	Group Average	Total Average	Individual Case	Group Average	Total Average
N/0.5	85%	82%	76%	80%	67%	57%
0.5/3	79%			60%		
N/3	81%			62%		
N/0.5 & 0.5/3	69%	76%		52%	59%	
0.5/3 & N/3	79%			53%		
N/0.5 & N/3	80%			72%		
All combined	55%	55%		17%	17%	

GO Enrichment Analysis on Filtered Genes

As shown in Fig. 3(a), a large portion of cScore and ECG mask-filtered DEGs were shared (73%) with DEGs obtained using P-values and FC values. Figure 2(b) and (c) show the top 10 enriched GO terms in the molecular function (MF) and biological process (BP) categories determined using g:Profiler (Kolberg et al., 2023), revealing that the cScore successfully excluded a large portion of genes related to inflammatory and immunological responses.

(a)



(b)

cScore & ECG mask only	
Ontology	GO Term
MF	protein-glutamine glutaminase activity
MF	binding
MF	protein-containing complex binding
MF	actin filament binding
MF	histone dopaminyltransferase activity
MF	histone serotonyltransferase activity
MF	peptide dopaminyltransferase activity
MF	peptide histaminyltransferase activity
MF	peptide serotonyltransferase activity
MF	interleukin-4 receptor activity

(c)

p value & up FC only	
Ontology	GO Term
BP	immune response
BP	DNA unwinding involved in DNA replication
BP	regulation of programmed cell death
BP	DNA replication initiation
MF	binding
MF	phosphorylation-dependent protein binding
BP	female meiotic nuclear division
MF	structural constituent of chromatin
BP	cellular response to UV-A
BP	keratinocyte activation

Fig. 3. Comparison between cScore & ECG masked DEGs of all combined dataset and p value & up regulated DEGs in N/0.5 & N/3 dataset. (a) Venn-diagram of DEGs filtered. (b) Top 10 GO Terms in MF and BP enriched from DEGs in cScore & ECG masked method only. (c) Top 10 GO Terms in MF and BP enriched from DEGs in p value & up regulation filtered method only.
(Note: MF = Molecular Function, BP = Biological Process)

Evaluation of Ranking in Filtered Genes.

Table 2 shows the ranks of genes related to the keyword sets put in. Genes were selected based on researches mainly focusing on serum biomarkers for brain damage (Korfias et al., 2009; Voznyuk et al., 2022). Glial fibrillary acidic protein (Gfap), myelin basic protein (Mbp), neuron specific enolase (Nse), microtubule associated protein tau (Mapt), and fatty acid-binding protein (FABP7) were grouped as non-inflammatory related genes. Interleukin 6 (Il6), S100 calcium-binding protein B (S100b), C-C motif chemokine ligand 5 (CCL5), tumor necrosis factor alpha (TNFα), heat shock protein family A (Hsp70) (Hspa-), suppressor of cytokine signaling 3 (Socs3), and tumor necrosis factor, alpha-induced protein 3 (Tnfaip3) were grouped as inflammatory related genes. It shows that ranking based on cScore effectively lowers the ranks for the majority of genes associated with inflammation and raises the ranks for genes that were not closely related to inflammation. The similarity between the results for cScore and FC with P-value cut-off can mainly be explained by the expression characteristic group avoidance score.

Table 2. Genes of interest ranked in whole library of genes. (Total number of unique gene being 20,774) "p val" column ranked by the adjusted p value of each gene from lowest to highest. "FC" column ranked by fold change. Any element that is absent in that ranking list is represented as "#" (continued)

Genes of interest		cScore	cScore (cScore>0)	cScore (ECG mask)	p val	FC	FC (p<0.05)
Group	Gene Name						
Non-Inflammatory	Gfap	108	108	1	71	360	174
	Mbp	337	337	#	11070	11174	#
	Nse(Eno2)	633	633	#	11393	10091	#
	Mapt	#	#	#	8282	14419	#
	Fabp7(B-Fabp)	2471	2471	#	2876	3343	#
Inflammatory	Il6	15337	#	#	386	41	#
	S100b	14994	#	#	6251	16864	#
	CCL5	15042	#	#	5830	901	#
	TNFα	15359	#	#	11851	4661	#
	Hspa1a	3398	3398	121	4	9	8

Table 2. Genes of interest ranked in whole library of genes. (Total number of unique gene being 20,774) "p val" column ranked by the adjusted p value of each gene from lowest to highest. "FC" column ranked by fold change. Any element that is absent in that ranking list is represented as "#"

Genes of interest		cScore	cScore (cScore>0)	cScore (ECG mask)	p val	FC	FC (p<0.05)
Group	Gene Name						
Inflammatory	Hspa1l	3191	3191	102	295	119	72
	Hspa1b	1018	1018	24	3	8	7
	Hspa2	1484	1980	#	811	1583	#
	Hspa4	10318	1484	#	9760	8248	#
	Hspa4l	1651	#	#	11382	14350	#
	Hspa5	14478	1651	#	2620	3227	#
	Hspa9	5274	#	#	8414	7624	#
	Hspa12a	2840	#	#	4802	17534	#
	Hspa12b	5702	2840	#	4809	7694	#
	Hspa13	6021	#	#	10621	11387	#
	Hspa14	14983	#	#	7464	5582	#
	Hspa8	#	#	#	8310	7520	#
	Hspa8-ps25	#	#	#	11776	6811	#
	Socs3	3469	3469	130	10	29	22
	Tnfaip3	15323	#	#	7138	4889	#

Effect of Extended Search in GO Graph

As stated previously, the user can define whether downward accumulation in the GO graph will affect only the terms with a hit or will affect other terms without a hit. We evaluate the effect of this option using the previously described dataset involving normal rats and rats with MCAO (0.5 h and 3 h).

An upset plot was generated by UpSetR package (Conway et al., 2017) as shown at Fig 4, while all the scored datasets shared 50% of filtered DEGs in common, there was a large difference of 88 genes (37%) between datasets obtained when setting extended search to TRUE or FALSE. This was much larger than the difference when applying the extended avoidance search (i.e., 16 genes or 7%). This suggests that the terms with a hit to avoidance keywords were well clustered and that keywords for searching might have been rather generic.

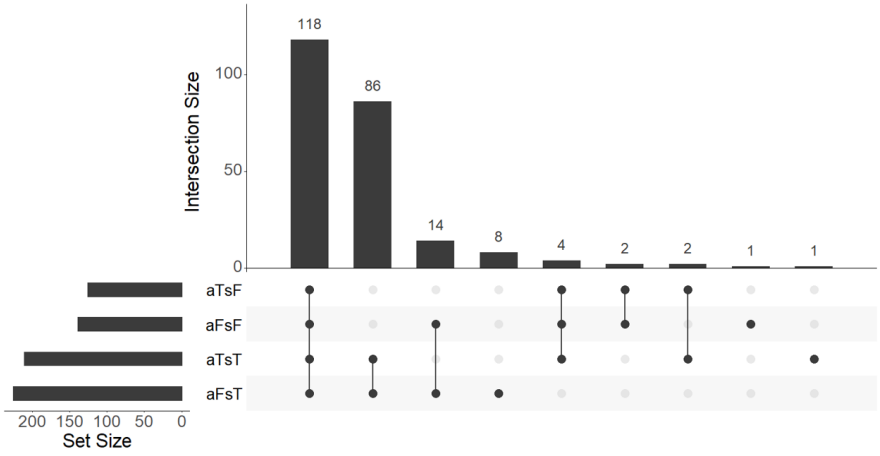


Fig. 4. Upset plot of cScore + ECG masked filtered DEGs. Each data set names represent if the GO graph downward accumulation was extended for each avoidance and search.
(a = avoidance, s = search, T = Extended Search True, F = Extended Search False

Enriched GO terms for genes at the aTsT and aFsT intersections are shown in Table 3. The extension of the search clearly resulted in higher scores for immune responses and inflammatory processes. Based on this result, where the extension of the search collides with avoidance or vice versa, it is advised to reevaluate the keywords used for network propagation.

Table 3. Result of GO enrichment analysis by g:Profiler (Liis Kolberg et al., 2023), on genes only in aTsT score set and aFsT score set. Only one term for each GO term cluster is shown.
(Note: MF = Molecular Function, BP = Biological Process, CC = Cellular Component)

cScore & ECG mask – aTsT score set & aFsT score set	
Ontology	GO Term
MF	structural constituent of chromatin
BP	immune system process
BP	regulation of epidermis development
BP	oxygen transport
BP	antigen processing and presentation of exogenous peptide antigen via MHC class II
BP	keratinocyte migration
BP	T cell activation involved in immune response
BP	innate immune response-activating signaling pathway
BP	response to interleukin-6
BP	osteoclast differentiation
CC	hemoglobin complex
CC	nucleosome

Discussions

In this study, a method for scoring genes based on keyword searches (GO terms) and network propagation was proposed and applied to a DEG dataset for a rat model of MCAO for comparison with traditional methods.

DEG analyses are one of the most comprehensive tools for gaining insight into the linkage between molecular biology and biological traits and have been in constant development since their emergence (Oshlack et al., 2010). Many normalization techniques and fitting methods have been studied to obtain more accurate results (Abbas-Aghababazadeh et al., 2018; Zyprych-Walczak et al., 2015). Advances in technology since the development of next-generation sequencing platforms have drastically increased the amount of data produced from each experiment (Chen et al., 2011; Day-Williams & Zeggini, 2011; Mortazavi et al., 2008). This requires a comprehensive strategy to analyze biological data; thus, a vast number of methods for enriching data have been developed (Maleki et al., 2020) and various databases have been built, such as the Gene Ontology database. These methods are effective for obtaining biological insights based on DEG data; however, it is difficult to gather information on significant DEGs that are not yet annotated (Khatri & Drăghici, 2005). Methods for gene prioritization fall into four major categories (Raj & Sreeja, 2018), including text mining, network-based, machine learning, and hybrid methods. However, methods to prioritize genes with the most probable biological relevance and genes involved in biological processes of interest to researchers is challenging. By utilizing portions of the framework used in gene prioritization, mainly Gene Ontology and PPI networks, it was possible to create a scoring method for keyword-based biases.

Nevertheless, several improvements to the proposed method should be considered. A key limitation is that the

method incorporates only a single database for each step: Gene Ontology for keyword matching and STRINGdb for network propagation. Because the scoring formula allows multiple score sets to be used in combination, it is possible to incorporate multiple annotation databases (Raj & Sreeja., 2018) for keyword searching, which will improve the quality of the generated score. Moreover, Li and Li (2012) showed that the construction and utilization of multigraphs and complex heterogeneous networks are effective for random-walk propagation, and applying the same concept for this specific usage by combining multiple networks will also improve the quality of the final score.

As shown in Fig. 3 and Table 3, the result of extended searching was heavily impacted by the keywords given, where the effect of extended searches in avoidance keywords was significantly weaker than that of extended searches for preferences. Therefore, the proposed method does not balance the effects of each keyword set.

Despite these limitations, the suggested method showed promising results for the original purpose of scoring based on keywords, while still retaining biological information to a certain extent. Further improvements should be made by incorporating additional databases and improving the robustness of the steps.

Conclusions and Implications

In this study, a keyword-based gene prioritization method combined with expression characteristic grouping was proposed. Keyword based search was performed through Gene Ontology graph, and propagated through protein-protein interaction network for compensating the incompleteness of gene annotation.

Its effectiveness was evaluated using DEG datasets for MCAO rats and normal rats, compared to the conventional method using fold change and value of significance. The result showed that proposed method is effective in biasing the score based on keywords. Furthermore, an improved version of this method would be effective for manually summarizing and analyzing DEG data with a set of interest.

Conflicts of Interest

The author(s) declared no conflicts of interest.

References

Abbas-Aghababazadeh, F., Li, Q., & Fridley, B. L. (2018). Comparison of normalization approaches for gene expression studies completed with high-throughput sequencing. PLOS ONE, 13, e0206312.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature genetics, 25, 25-29.

Baldi, P., & Long, A. D. (2001). A Bayesian framework for the analysis of microarray expression data: regularized t -test and statistical inferences of gene changes. Bioinformatics (Oxford, England), 17, 509–519.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful

approach to multiple testing. Journal of the Royal Statistical Society, 57289–300
Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics, 30, 2114–2120.
Carlson, M. (2023). GO.db: A Set of Annotation Maps Describing The Entire Gene Ontology. R Package Version 3.18.0.
Chen, G., Wang, C., & Shi, T. (2011). Overview of available methods for diverse RNA-Seq data analyses. Science China Life Sciences, 54, 1121–1128.
Chen, J., Aronow, B. J., & Jegga, A. G. (2009). Disease candidate gene identification and prioritization using protein interaction networks. BMC Bioinformatics, 10, 73.
Choi, B. I., Park, D., Lee, S. H., Bae, D. K., Yang, G., Yang, Y. H., Kim, T. K., Choi, E. K., Lee, H. J., Choi, K. C., Nahm, S. S., & Kim, Y. B. (2012). Neurobehavioural deficits correlate with the cerebral infarction volume of stroke animals: A comparative study on ischaemia-reperfusion and photothrombosis models. Environmental Toxicology and Pharmacology, 33, 60-69.
Claverie J. M. (1999). Computational methods for the identification of differential and coordinated gene expression. Human molecular genetics, 8, 1821–1832.
Conway, J. R., Lex, A., & Gehlenborg, N. (2017). UpSetR: An R package for the visualization of intersecting sets and their properties. Bioinformatics, 33, 2938–2940.
Day-Williams, A. G., & Zeggini, E. (2011). The effect of next-generation sequencing technology on complex trait research. European journal of clinical investigation, 41, 561–567.
Dirmeier, S. (2018). diffusr: Network Diffusion Algorithms. R Package Version 0.1.4, <https://CRAN.R-project.org/package=diffusr>.
Grote, S. (2023). GOfuncR: Gene ontology enrichment using FUNC. R package version 1.22.0, https://10.18129/B9.bioc.GOfuncR,
Hinderer, E. W., 3rd, Flight, R. M., Dubey, R., MacLeod, J. N., & Moseley, H. N. B. (2019). Advances in gene ontology utilization improve statistical power of annotation enrichment. PloS One, 14, e0220728.
Khatri, P., & Drăghici, S. (2005). Ontological analysis of gene expression data: current tools, limitations, and open problems. Bioinformatics, 21, 3587–3595.
Kim, J., Shin, K., Cha, Y., Ban, Y. H., Park, S. K., Jeong, H. S., Park, D., Choi, E. K., & Kim, Y. B. (2020). Neuroprotective effects of human neural stem cells over-expressing choline acetyltransferase in a middle cerebral artery occlusion model. Journal of Chemical Neuroanatomy, 103, 101730.
Kolberg, L., Raudvere, U., Kuzmin, I., Adler, P., Vilo, J., & Peterson, H. (2023). G:Profiler—interoperable web service for functional enrichment analysis and gene identifier mapping (2023 update). Nucleic Acids Research, 51, W207–W212.
Korfias, S., Papadimitriou, A., Stranjalis, G., Bakoula, C., Daskalakis, G., Antsaklis, A., & Sakas, D. E. (2009). Serum biochemical markers of brain injury. Mini Reviews in Medicinal Chemistry, 9, 227–234.
Li, Y., & Li, J. (2012). Disease gene identification by random walk on multigraphs merging heterogeneous genomic and phenotype data. BMC Genomics 13, Suppl 7, S27.
Longa, E. Z., Weinstein, P. R., Carlson, S., & Cummins, R. (1989). Reversible middle cerebral artery occlusion without craniectomy in rats. Stroke, 20, 84-91.
Maleki, F., Ovens, K., Hogan, D. J., & Kusalik, A. J. (2020). Gene Set Analysis: Challenges, Opportunities, and Future Research. Frontiers in Genetics, 11, 654.
Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., & Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature Methods, 5, 621–628.
Oshlack, A., Robinson, M. D., & Young, M. D. (2010). From RNA-seq reads to differential expression results. Genome Biology, 11, 220.

Oti, M., Snel, B., Huynen, M. A., & Brunner, H. G. (2006). Predicting disease genes using protein–protein interactions. Journal of Medical Genetics, 43, 691-698.

Park, D., Joo, S. S., Lee, H. J., Choi, K. C., Kim, S. U., & Kim, Y. B. (2012). Microtubule-associated protein 2, an early blood marker of ischemic brain injury. Journal of Neuroscience Research, 90, 461-467.

Patterson, T. A., Lobenhofer, E. K., Fulmer-Smentek, S. B., Collins, P. J., Chu, T. M., Bao, W., … Wolfinger, R. D. (2006). Performance comparison of one-color and two-color platforms within the MicroArray Quality Control (MAQC) project. Nature Biotechnology, 24, 1140–1150.

Peart, M. J., Smyth, G. K., van Laar, R. K., Bowtell, D. D., Richon, V. M., Marks, P. A., Holloway, A. J., & Johnstone, R. W. (2005). Identification and functional significance of genes regulated by structurally different histone deacetylase inhibitors. Proceedings of the National Academy of Sciences of the United States of America, 102, 3697–3702.

Pervez, M. T., Hasnain, M. J. U., Abbas, S. H., Moustafa, M. F., Aslam, N., & Shah, S. S. M. (2022). A Comprehensive Review of Performance of Next-Generation Sequencing Platforms. BioMed research international, 2022, 3457806. https://doi.org/10.1155/2022/3457806

R Core Team. (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. <https://www.R-project.org/>.

Raj, M. R., & Sreeja, A. (2018). Analysis of computational gene prioritization approaches. Procedia Computer Science, 143, 395-410.

Raouf, A., Zhao, Y., To, K., Stingl, J., Delaney, A., Barbara, M., Iscove, N., Jones, S., McKinney, S., Emerman, J., Aparicio, S., Marra, M., & Eaves, C. (2008). Transcriptome analysis of the normal human mammary cell commitment and differentiation process. Cell Stem Cell, 3, 109–118.

Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139-140.

Schmid-Elsaesser, R., Zausinger, S., Hungerhuber, E., Baethmann, A., & Reulen, H. J. (1998). Monotherapy with dextromethorphan or tirilazad–but not a combination of both–improves outcome after transient focal cerebral ischemia in rats. Exp. Brain Res, 122, 121–127.

Stark, R., Grzelak, M., & Hadfield, J. (2019). RNA sequencing: the teenage years. Nature Reviews Genetics, 20, 631–656.

Szklarczyk, D., Kirsch, R., Koutrouli, M., Nastou, K., Mehryary, F., Hachilif, R., Gable, A. L., Fang, T., Doncheva, N. T., Pyysalo, S., Bork, P., Jensen, L. J., & von Mering, C. (2023). The STRING database in 2023: protein-protein association networks and functional enrichment analyses for any sequenced genome of interest. Nucleic Acids Research, 51, D638–D646.

The Gene Ontology Consortium. (2023). The Gene Ontology knowledgebase in 2023. Genetics, 224(1), iyad031.

Urbanek, S. (2023). Fastmatch: Fast 'match()' Function. R Package Version 1.1-4, <https://CRAN.R-project.org/package=fastmatch>.

Voznyuk, I. A., Pivovarova, L. P., Gogoleva, E. A., Osipova, I. V., Ariskina, O. B., Morozova, E. M., Chernyavsky, I. V., & Markelova, E. V. (2022). Biomarkery povrezhdeniya mozga i vospaleniya u patsientov s ostroi tserebral'noi ishemiei [Biomarkers of brain damage and inflammation in patients with acute cerebral ischemia]. Zhurnal nevrologii i psikhiatrii imeni S.S. Korsakova, 122, 54–60.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., … Yutani, H. (2019). Welcome to the tidyverse. Journal of Open Source Software, 4, 1686.

Young, M. D., McCarthy, D. J., Wakefield, M. J., Smyth, G.K., Oshlack, A., & Robinson, M.D. (2012). Differential Expression for RNA Sequencing (RNA-Seq) Data: Mapping, Summarization, Statistical Analysis, and Experimental Design. In: N. Rodríguez-Ezpeleta., M. Hackenberg., & A. Aransay. (Eds) Bioinformatics for High Throughput Sequencing. Springer, New York, NY. 169-190.

Zyprych-Walczak, J., Szabelska, A., Handschuh, L., Górczak, K., Klamecka, K., Figlerowicz, M., & Siatkowski, I. (2015). The Impact of Normalization Methods on RNA-Seq Data Analysis. BioMed Research International, 2015, Article ID 621690.

Authors' Information

Kim, Songhwan: Korea National University of Education, Undergraduate Student, First Author

Park, Dongsun: Korea National University of Education, Professor, Corresponding Author

Appendix

Source Code Used

```
#------------------------------------------------------------------------------------------------
# Package initializing functions
#------------------------------------------------------------------------------------------------
{
 GenericP <- function(x){
  for( i in x ){
   if( ! require( i , character.only = TRUE ) ){
    install.packages( i , dependencies = TRUE )
    require( i , character.only = TRUE )
   }
  }
 }
 BiocP <- function(x){
  for( i in x ){
   if( ! require( i , character.only = TRUE ) ){
    BiocManager::install(i)
    require( i , character.only = TRUE )
   }
  }
 }

}
#------------------------------------------------------------------------------------------------
# Load Generic Packages
#------------------------------------------------------------------------------------------------
```

```
{
 message("[Loading generic packages]")
 GenericP(c("tidyverse","stringr","diffusr","fastmatch","BiocManager"))
}
#---------------------------------------------------------------------------------------------------
# Load Bioconductor Packages
#---------------------------------------------------------------------------------------------------
{
 message("[Loading bioconductor packages]")
 BiocP(c("GO.db","GOfuncR","STRINGdb"))
}
#---------------------------------------------------------------------------------------------------
# Initialize Functions
#---------------------------------------------------------------------------------------------------
{
 message("[Initializing custom functions]")
  {
   tknChar<-function(type,value,input,current){
    if(value==substr(input,current,current)) return(list(1,list("type"=type,"value"=value)))
    else return(list(0,NULL))
   }
   tknPO<-function(input,current){
    tknChar("paren","(",input,current)
   }
   tknPC<-function(input,current){
    tknChar("paren",")",input,current)
   }
   tknOpO<-function(input,current){
    tknChar("operation","|",input,current)
   }
   tknOpA<-function(input,current){
    tknChar("operation","&",input,current)
   }
   tknOpN<-function(input,current){
    tknChar("operation","!",input,current)
   }
```

```
tknStr<-function(input,current){
 if(substr(input,current,current)==""){
  v=""
  n=1
  c=substr(input,current+n,current+n)
  while(c!=""){
   if(c=="")stop("unterminated string")
   v=paste0(v,c)
   n=n+1
   c=substr(input,current+n,current+n)
  }
  return(list(n+1,list("type"="string","value"=v)))
 }
 return(list(0,NULL))
}
skpWS<-function(input,current){
 if(str_detect(substr(input,current,current),regex("\\s")))return(list(1,NULL))
 else return(list(0,NULL))
}
tknLs<-c("po"=tknPO,"pc"=tknPC,"opO"=tknOpO,"opA"=tknOpA,"opN"=tknOpN,"str"=tknStr,"ws"=skpWS)
tkn<-function(input){
 i=1
 tkns=list(list("type"="paren","value"="("))
 while(i<=str_length(input)){
  tknzd=F
  c=substr(input,i,i)
  for(j in 1:length(tknLs)){
   if(tknzd)break
   n=tknLs[[j]](input,i)
   if(n[[1]]!=0){
    tknzd=T
    i=i+n[[1]]
   }
   if(!is.null(n[[2]])){
    tkns[[length(tkns)+1]]=n[[2]]
   }
```

```
    }
    if(!tknzd){
     stop(paste0("Unexpected character : ",c))
    }
   }
   tkns[[length(tkns)+1]]=list("type"="paren","value"=")")
   return(tkns)
  }
}
inToPost<-function(tokens){
 t=0
 tkns=list()
 stk=list()
 for(i in 1:length(tokens)){
  tkn=tokens[[i]]
  if(tkn$type=="string")tkns[[length(tkns)+1]]=tkn
  if(tkn$type=="paren"){
   if(tkn$value=="("){
    stk[[t+1]]=tkn
    t=t+1
   }
   else{
    while(!(stk[[t]]$type=="paren"&stk[[t]]$value=="(")){
     tkns[[length(tkns)+1]]=stk[[t]]
     t=t-1
    }
    t=t-1
   }
  }
  if(tkn$type=="operation"){
   if(tkn$value=="!"){
    stk[[t+1]]=tkn
    t=t+1
   }
   if(tkn$value=="&"|tkn$value=="|"){
    if(t==0){
```

```
      stk[[t+1]]=tkn
     }
     else{
      while(stk[[t]]$value!="("){
       if(stk[[t]]$value=="!"){
        tkns[[length(tkns)+1]]=stk[[t]]
        t=t-1
       }
       if(stk[[t]]$value=="&"|stk[[t]]$value=="|"){
        tkns[[length(tkns)+1]]=stk[[t]]
        t=t-1
        break
       }
      }
      stk[[t+1]]=tkn
      t=t+1
     }
    }
   }
  }
  if(t!=0) for(i in 1:t){
   tkns[[length(tkns)+1]]=stk[[t-i+1]]
  }
  return(tkns)
}
tknCalc<-function(tokens,ilist){
  t=0
  li=1
  stk=list()
  for(i in 1:length(tokens)){
   tkn=tokens[[i]]
   if(tkn$type=="string"){
    t=t+1
    stk[[t]]=ilist[[li]]
    li=li+1
   }
```

```
  if(tkn$type=="operation"){
   if(tkn$value=="!"){
    stk[[t]]=!stk[[t]]
   }
   else{
    if(tkn$value=="&"){
     stk[[t-1]]=stk[[t]]&stk[[t-1]]
    }
    else{
     stk[[t-1]]=stk[[t]]|stk[[t-1]]
    }
   }
  }
 }
 return(stk[[1]])
}
`%fin%` <- function(x, table) {
 fmatch(x, table, nomatch = 0L) > 0L
}
GOoboToTermDF<-function(file){
 pTable<-data.frame(
  "go_id"=c("tracker"),
  "name"=c("tracker"),
  "namespace"=c("tracker"),
  "def"=c("tracker"),
  "synonym"=I(list(c("tracker"))),
  "is_a"=I(list(c("tracker"))),
  "relationship"=I(list(c("tracker"))),
  "is_obsolete"=c(F),
  "is_root"=c(F))
 root<-c("biological_process","cellular_component","molecular_function")
 c=1
 pb=txtProgressBar(min=1,initial=1,max=length(file),style=3)
 for(i in 1:length(file)){
  if(file[i]=="[Term]"){
   pTable<-rbind(pTable,data.frame(
```

```
  "go_id"=c(""),
  "name"=c(""),
  "namespace"=c(""),
  "def"=c(""),
  "synonym"=I(list(NULL)),
  "is_a"=I(list(NULL)),
  "relationship"=I(list(NULL)),
  "is_obsolete"=c(F),
  "is_root"=c(F)))
 c=c+1
}
else{
 if(startsWith(file[i],"i")){
  if(startsWith(file[i],"is_a:")){
   pTable$is_a[[c]]=c(pTable$is_a[[c]],substr(file[i],7,16))

  }else if(startsWith(file[i],"id:")){
   pTable$go_id[c]=substr(file[i],5,nchar(file[i]))

  }else if(startsWith(file[i],"is_obsolete:")){
   pTable$is_obsolete[c]=T

  }
 }else if(startsWith(file[i],"synonym:")){
  if(grepl("\" EXACT \\[",file[i])){
   pTable$synonym[[c]]=c(pTable$synonym[[c]],list(gsub("\"[^\"]+$","",gsub("^[^\"]+\"","",file[i]))))
  }

 }else if(startsWith(file[i],"name")){
  if(startsWith(file[i],"name:")){
   pTable$name[c]=substr(file[i],7,nchar(file[i]))
   if(pTable$name[c]%in%root)pTable$is_root[c]=T
  }
  else{
   pTable$namespace[c]=substr(file[i],12,nchar(file[i]))
  }
```

```
    }else if(startsWith(file[i],"def:")){
     pTable$def[c]=gsub("\"[^\"]+$","",gsub("^[^\"]+\"","",file[i]))

    }else if(startsWith(file[i],"relationship:")){
     pTable$relationship[[c]]=
      c(pTable$relationship[[c]],
       list(c(
        gsub(" GO:.*","",
          gsub("relationship: ","",file[i])),
        gsub(" ! .*","",
          gsub("^\\w+: [^ ]+ ","",file[i])))))
     }
    }
    if(i%%500==0){
     setTxtProgressBar(pb,i)
    }
   }
   close(pb)
   gc()
   return(pTable)
  }
  colnamesFormat<-function(cvec){
   temp=cvec
   temp[grepl("^\\d",temp)]=paste0("X",temp[grepl("^^\\d",temp)])
   temp<-gsub("/",".",temp)
   return(temp)
  }
}
#-----------------------------------------------------------------------------------------------
# Check & Read External Files
#-----------------------------------------------------------------------------------------------
{
  message("[Reading in required external files]")

  if(file.exists("bin/annotRef")){
```

```
 load("bin/annotRef")
}else{
 if(!file.exists("rsc/rgd.gaf.gz")){
  download.file(
   "http://current.geneontology.org/annotations/rgd.gaf.gz",
   "rsc/rgd.gaf.gz",
   cacheOK=F,
   mode="wb"
  )
 }
 annotRef<-read.table("rsc/rgd.gaf.gz",sep="\t",comment.char="!",header=F)
 save(annotRef,file="bin/annotRef")
}
if(file.exists("bin/GOdf")){
 load("bin/GOdf")
}else{
 if(!file.exists("goDir/go-basic.obo")){
  download.file(
   "http://current.geneontology.org/ontology/go-basic.obo",
   "goDir/go-basic.obo",
   cacheOK=F,
   mode="wb"
  )
 }
 GOdf<-GOoboToTermDF(read_lines("goDir/go-basic.obo"))
 save(GOdf,file="bin/GOdf")
}
if(file.exists("bin/geneTable")){
 load("bin/geneTable")
}else{
 if(!file.exists("rsc/GENES.RAT.txt")){
  download.file(
   "https://download.rgd.mcw.edu/data_release/GENES.RAT.txt",
   "rsc/GENES.RAT.txt",
   cacheOK=F,
   mode="wb"
```

```
    )
   }
   geneTable<-read.table(
    "rsc/GENES.RAT.txt",
    sep="\t",
    comment.char="#",
    header=T,
    fill=NA,
    quote=""
   )[,c("GENE_RGD_ID","SYMBOL","NAME","GENE_DESC","NCBI_GENE_ID","ENSEMBL_ID")]
   rownames(geneTable)<-geneTable$GENE_RGD_ID
   save(geneTable,file="bin/geneTable")
  }
}
#-----------------------------------------------------------------------------------------------
# User Input Field
#-----------------------------------------------------------------------------------------------
{
 message("[Reading user inputs]")

 # U,N,D...
 remGroup<-c("NNN","NND","NDN","NDD","DNN","DND","DDN","DDD")
 lookupGroup<-c("UUU","UNU")

 # 0~1 (restart probability if markov random walk with restart is desired)
 sRWr=0.5
 aRWr=0.5
 remRWr=0.5

 iteration=100000
 rThresh=1e-10

 # Weights to be applied to each rankings in the calculation of combined score (cScore)
 calcWeight<-list("s"=1,"a"=1,"rem"=1)

 # Keyword sets
```

```
avoidance<-
 '"Inflamma"|"inflamma"|
 "Immun"|"immun"|
 "Interleukin"|"interleukin"|
 "Lymphocyte"|"lymphocyte"|
 "sarcoma"|"oncogene"|"cancer"|"lymphoma"'
searching<-'"Neuro"|"neuro"|"Synap"|"synap"|"Nervous"|"nervous"'

# Boolean for extending downward accumulation to terms without any hit
extend<-list("search"=F,"avoidance"=F)

# 1=True, 0=False. Selected to define the fields for search to be done.
matchRange<-list("Name"=1,"Definition"=1,"Synonym"=1)

# "N","A","S" (only applied when both avoidBool and searchBool is true)
AS_bias="N"

# Stringdb initialization
string_db<-STRINGdb$new(
 version="11.5",
 species=10116,
 score_threshold=700,
 network_type="full",
 input_directory="./rsc/string_db"
 )
refLib<-base::unique(geneTable$NCBI_GENE_ID)

# Main data file
dataF<-read.csv("proc.csv")

# Gene ids should be in ENTREZ id
geneCol<-c("Gene_ID")

# Colnames containing fold change data (in the same order with significance)
FCcolnames<-c("X0_5_Nor_logFC","X3_X0_5_logFC","X3_Nor_logFC")
FCThresh<-0
```

```
 # Colnames contating significance data (in the same order with fold-change)
 sigvalcolnames<-c("X0_5_Nor_pAdj_bh","X3_X0_5_pAdj_bh","X3_Nor_pAdj_bh")
 sigThresh<-0.05

 LPfilt<-c()
 LPfiltThrs<-c()
 HPfilt<-c()
 HPfiltThrs<-c()
}
#-----------------------------------------------------------------------------------------------------
# Validate user inputs
#-----------------------------------------------------------------------------------------------------
{
 geneCol<-colnamesFormat(geneCol)
 FCcolnames<-colnamesFormat(FCcolnames)
 sigvalcolnames<-colnamesFormat(sigvalcolnames)
 LPfilt<-colnamesFormat(LPfilt)
 HPfilt<-colnamesFormat(HPfilt)

 if(length(FCcolnames)==0)stop("You need at least 1 comparison set")
 if(length(FCcolnames)!=length(sigvalcolnames))stop("FC columns and p-value columns length not matching")
 temp<-c(FCcolnames,sigvalcolnames,LPfilt,geneCol)
 if(any(!temp%fin%colnames(dataF))){
        stop(paste("Following columns aren't defined : ",temp[!temp%fin%colnames(dataF)]))
 }
}
#-----------------------------------------------------------------------------------------------------
# Attach STRINGdb info
#-----------------------------------------------------------------------------------------------------
{
 message("[Tieing string DB]")
 refLib<-c(refLib,as.integer(dataF[!dataF$Gene_ID%fin%refLib,geneCol]))
 refLib<-set_names(as.data.frame(refLib),"ENTREZ")
 wLib<-string_db$map(refLib,"ENTREZ",removeUnmappedRows=T)
 rownames(wLib)<-paste0(wLib$ENTREZ,"|",wLib$STRING_id)
```

```
}
#-----------------------------------------------------------------------------------------------------
# GO Term Table initialization
#-----------------------------------------------------------------------------------------------------
{
 message("[GO table initializing]")
 if(file.exists("bin/TermLs")){
  load("bin/TermLs",)
 }else{
  TermLs<-GOdf[!GOdf$is_obsolete,c(1,2,3,4,5)]
  TermLs<-TermLs[-1,]
  TermLs<-TermLs[-nrow(TermLs),]
  colnames(TermLs)<-c("go_id","Term","Ontology","Definition","Synonym")
  pb=txtProgressBar(min=1,initial=1,max=nrow(TermLs),style=3)
  for(i in 1:nrow(TermLs)){
   if(is.null(TermLs$Synonym[[i]]))TermLs$Synonym[[i]]=""
   else TermLs$Synonym[[i]]=paste(unlist(TermLs$Synonym[[i]]),collapse="\\")
   TermLs$Ontology[i]<-
        c("MF","CC","BP")[TermLs$Ontology[i]==c("molecular_function","cellular_component","biological_
process")]
   setTxtProgressBar(pb,i)
  }
  close(pb)
  gc()
  rownames(TermLs)<-TermLs$go_id
  TermLs$rgd<-c(I(list(NA)))
  tempC<-c(
        "enalbes",
        "contributes_to",
        "acts_upstream_of",
        "involved_in",
        "located_in",
        "part_of",
        "is_active_in"
        )
  pb=txtProgressBar(min=1,initial=1,max=nrow(annotRef),style=3)
```

```
  for(i in 1:nrow(annotRef)){
   if(any(startsWith(annotRef$V4[i],tempC))){
    TermLs[annotRef$V5[i],]$rgd[[1]]=c(TermLs[annotRef$V5[i],]$rgd[[1]],list(annotRef$V2[i]))
   }
   setTxtProgressBar(pb,i)
  }
  close(pb)

  pb=txtProgressBar(min=1,initial=1,max=nrow(TermLs),style=3)
  for(i in 1:nrow(TermLs)){
   if(!is.na(TermLs$rgd[i])){
    TermLs$rgd[[i]]<-base::unique(unlist(TermLs$rgd[[i]])[-1])
   }
   setTxtProgressBar(pb,i)
  }
  close(pb)

  TermLs$genes<-c(I(list(NA)))
  pb=txtProgressBar(min=1,initial=1,max=nrow(TermLs),style=3)
  for(i in 1:nrow(TermLs)){
   if(!is.na(TermLs$rgd[i])){
    TermLs$genes[[i]]<-geneTable[TermLs$rgd[[i]],]$NCBI_GENE_ID
   }
   setTxtProgressBar(pb,i)
  }
  close(pb)

  save(TermLs,file="bin/TermLs")
 }
}
#-----------------------------------------------------------------------------------------------------
# Search & Hit, Downward Accumulation in GO graph
#-----------------------------------------------------------------------------------------------------
{
 message("[Searching terms]")
 avoidBool<-F
```

```
if(str_length(avoidance)!=0){
 aTkns<-inToPost(tkn(avoidance))
 aStrV<-c()
 for(i in 1:length(aTkns)){
  if(aTkns[[i]]$type=="string")aStrV<-c(aStrV,aTkns[[i]]$value)
 }
 if(length(aStrV)==0)warning("Avoidance term search can't be performed without any terms as arguments")
 else{
  aHitLs<-list()
  for(i in 1:length(aStrV)){
   aHitLs[[i]]=
    (str_detect(TermLs$Term,aStrV[i])&matchRange$Name)|
    (str_detect(TermLs$Definition,aStrV[i])&matchRange$Definition)|
    (str_detect(TermLs$Synonym,aStrV[i])&matchRange$Synonym)
  }
  TermLs$aHit<-tknCalc(aTkns,aHitLs)

  TermLs$aScore<-as.integer(TermLs$aHit)

  message("GO graph downward propagation score accumulating for avoidance")
  temp01<-TermLs[TermLs$aHit,]%>%
   distinct_at(.vars=c("go_id"))
  pb<-txtProgressBar(min=0,max=nrow(temp01),initial=0,style=3)
  temp02<-get_child_nodes(temp01$go_id,godir="goDir")
  for(i in 1:nrow(temp01)){
   temp<-temp02[temp02$parent_go_id==temp01$go_id[i],]
   if(nrow(temp)==1){setTxtProgressBar(pb,i);next}
   TermLs$aScore[TermLs$go_id%fin%temp$child_go_id[-1]]=
    TermLs$aScore[TermLs$go_id%fin%temp$child_go_id[-1]]+
    sum(temp$child_go_id[-1]%fin%temp01$go_id)/(nrow(temp)-1)*
    (extend$avoidance|TermLs$aHit[TermLs$go_id%fin%temp$child_go_id[-1]])
   setTxtProgressBar(pb,i)
  }
  close(pb)
  rm(temp,temp01,temp02)
```

```
  aTermLs<-TermLs[TermLs$aHit|(TermLs$aScore!=0&extend$avoidance),]
  aTermLs<-aTermLs[!is.na(aTermLs$genes),]
  aGenes<-character(0)
  for(i in 1:nrow(aTermLs)){
   aTermLs$genes[[i]]<-aTermLs$genes[[i]][!is.na(aTermLs$genes[[i]])]
   aGenes=c(aGenes,aTermLs$genes[[i]])
   aGenes=base::unique(aGenes)
  }
  aGenes<-data.frame("ENTREZ"=aGenes,"aScore"=0)
  rownames(aGenes)<-aGenes$ENTREZ
  for(i in 1:nrow(aTermLs)){
   aGenes[as.character(aTermLs$genes[[i]]),]$aScore=
                max(aGenes[as.character(aTermLs$genes[[i]]),]$aScore,aTermLs$aScore[i])
  }
  aGenes<-aGenes%>%
   arrange(ENTREZ)
  amGenes<-aGenes[aGenes$ENTREZ%fin%wLib$ENTREZ,]
  aumGenes<-aGenes[!aGenes$ENTREZ%fin%wLib$ENTREZ,]
  wLib<-wLib%>%
   mutate(aHit=ENTREZ%in%amGenes$ENTREZ)%>%
   left_join(amGenes,by="ENTREZ")
  wLib$aScore[is.na(wLib$aScore)]=0
  avoidBool<-T
 }
}
searchBool<-F
if(str_length(searching)!=0){
 sTkns<-inToPost(tkn(searching))
 sStrV<-c()
 for(i in 1:length(sTkns)){
  if(sTkns[[i]]$type=="string")sStrV<-c(sStrV,sTkns[[i]]$value)
 }
 if(length(sStrV)==0)warning("Searching term search can't be performed without any terms as arguments")
 else{
  sHitLs<-list()
  for(i in 1:length(sStrV)){
```

```
 sHitLs[[i]]=
  (str_detect(TermLs$Term,sStrV[i])&matchRange$Name)|
  (str_detect(TermLs$Definition,sStrV[i])&matchRange$Definition)|
  (str_detect(TermLs$Synonym,sStrV[i])&matchRange$Synonym)
}
TermLs$sHit<-tknCalc(sTkns,sHitLs)

TermLs$sScore<-as.integer(TermLs$sHit)

message("GO graph downward propagation score accumulating for searching")
temp01<-TermLs[TermLs$sHit,]%>%
 distinct_at(.vars=c("go_id"))
temp02<-get_child_nodes(temp01$go_id,godir="goDir")
pb<-txtProgressBar(min=0,max=nrow(temp01),initial=0,style=3)
for(i in 1:nrow(temp01)){
 temp<-temp02[temp02$parent_go_id==temp01$go_id[i],]
 TermLs$sScore[TermLs$go_id%fin%temp$child_go_id[-1]]=
  TermLs$sScore[TermLs$go_id%fin%temp$child_go_id[-1]]+
  sum(temp$child_go_id[-1]%fin%temp01$go_id)/(nrow(temp)-1)*
  (extend$search|TermLs$sHit[TermLs$go_id%fin%temp$child_go_id[-1]])
 setTxtProgressBar(pb,i)
}
close(pb)
rm(temp,temp01,temp02)

sTermLs<-TermLs[TermLs$sHit|(TermLs$sScore!=0&extend$search),]
sTermLs<-sTermLs[!is.na(sTermLs$genes),]
sGenes<-character(0)
for(i in 1:nrow(sTermLs)){
 sTermLs$genes[[i]]<-sTermLs$genes[[i]][!is.na(sTermLs$genes[[i]])]
 sGenes=c(sGenes,sTermLs$genes[[i]])
 sGenes=base::unique(sGenes)
}
sGenes<-data.frame("ENTREZ"=sGenes,"sScore"=0)
rownames(sGenes)<-sGenes$ENTREZ
for(i in 1:nrow(sTermLs)){
```

```
      sGenes[as.character(sTermLs$genes[[i]]),]$sScore=
                    max(sGenes[as.character(sTermLs$genes[[i]]),]$sScore,sTermLs$sScore[i])
    }
    sGenes<-sGenes%>%
     arrange(ENTREZ)
    smGenes<-sGenes[sGenes$ENTREZ%fin%wLib$ENTREZ,]
    sumGenes<-sGenes[!sGenes$ENTREZ%fin%wLib$ENTREZ,]
    wLib<-wLib%>%
     mutate(sHit=ENTREZ%in%smGenes$ENTREZ)%>%
     left_join(smGenes,by="ENTREZ")
    wLib$sScore[is.na(wLib$sScore)]=0
    searchBool<-T
   }
  }

  if(AS_bias!="N"&avoidBool&searchBool){
   if(AS_bias=="A"){
    wLib$sHit[wLib$aHit]=F
   }
   if(AS_bias=="S"){
    wLib$aHit[wLib$sHit]=F
   }
  }
}
#-----------------------------------------------------------------------------------------------------
# Map genes, Assign ECG
#-----------------------------------------------------------------------------------------------------
{
 message("[Filtering/Tieing data]")
 dataF<-dataF[!duplicated(dataF[,geneCol]),]
 dataF<-dataF%>%
  mutate(contain="")
 temp<-data.frame(matrix(ncol=4))[-1,]
 pb<-txtProgressBar(min=0,max=nrow(dataF),initial=0,style=3)
 for(i in 1:nrow(dataF)){
  temp=
```

```
        rbind(
         temp,
                geneTable[
                        geneTable$NCBI_GENE_ID%fin%dataF[i,geneCol],
                        c("NCBI_GENE_ID","SYMBOL","NAME","GENE_DESC")
                        ]
                )
 setTxtProgressBar(pb,i)
}
close(pb)
colnames(temp)<-c(geneCol,"SYMBOL","NAME","DESCRIPTION")
dataF<-dataF%>%
 left_join(temp,by=geneCol)

dataF<-dataF[,c(1,(ncol(dataF)-2):ncol(dataF),2:(ncol(dataF)-3))]

for(i in 1:length(FCcolnames)){
 dataF$contain=paste0(dataF$contain,case_when(
  dataF[,FCcolnames[i]]>(FCThresh)&dataF[,sigvalcolnames[i]]<=sigThresh ~ "U",
  dataF[,FCcolnames[i]]<(-FCThresh)&dataF[,sigvalcolnames[i]]<=sigThresh ~ "D",
  dataF[,sigvalcolnames[i]]>sigThresh|abs(dataF[,FCcolnames[i]])<=FCThresh ~ "N",
  .default="_"
 ))
}
mGs<-string_db$map(dataF,geneCol,removeUnmappedRows=TRUE)
unmGs<-dataF[!(dataF[,geneCol]%in%mGs[,geneCol]),]
rownames(unmGs)<-unmGs[,geneCol]
rownames(mGs)<-paste0(mGs[,geneCol],"|",mGs$STRING_id)
colnames(unmGs)[colnames(unmGs)==geneCol]<-"ENTREZ"
colnames(mGs)[colnames(mGs)==geneCol]<-"ENTREZ"

if(avoidBool){
 aumGenes<-aumGenes[aumGenes$ENTREZ%fin%unmGs$ENTREZ,]
 aumGenes$ENTREZ<-as.integer(aumGenes$ENTREZ)
 unmGs$ahit<-unmGs$ENTREZ%fin%aumGenes$ENTREZ
 unmGs<-unmGs%>%
```

```
    left_join(aumGenes,by="ENTREZ")
  unmGs$aScore[is.na(unmGs$aScore)]=0
  unmGs$aScore<--unmGs$aScore
 }
 if(searchBool){
  sumGenes<-sumGenes[sumGenes$ENTREZ%fin%unmGs$ENTREZ,]
  sumGenes$ENTREZ<-as.integer(sumGenes$ENTREZ)
  unmGs$shit<-unmGs$ENTREZ%fin%sumGenes$ENTREZ
  unmGs<-unmGs%>%
   left_join(sumGenes,by="ENTREZ")
  unmGs$sScore[is.na(unmGs$sScore)]=0
 }
}
#-----------------------------------------------------------------------------------------------------
# Matrix Construction for network propagation
#-----------------------------------------------------------------------------------------------------
{
 print("[Referance library-wise search network matrix initization]")
 netMat<-matrix(rep(0,nrow(wLib)^2),nrow=nrow(wLib),ncol=nrow(wLib))
 colnames(netMat)<-wLib$STRING_id
 rownames(netMat)<-paste0("X",wLib$STRING_id)

 print("Neighbor ID in list check")
 pb<-txtProgressBar(min=0,max=nrow(wLib),initial=0,style=3)
 temp01<-data.frame(matrix(character(),ncol=1,nrow=0))
 checkLs<-list()
 colnames(temp01)<-c("string")
 for(i in 1:nrow(wLib)){
  checkLs[[i]]=string_db$get_neighbors(c(wLib$STRING_id[i]))
  if(length(checkLs)!=i)checkLs[[i]]=NA
  temp01<-temp01%>%
   add_row(string=checkLs[[i]])%>%
   distinct_at(.vars=c("string"))
  setTxtProgressBar(pb,i)
 }
 close(pb)
```

```
  temp01<-cbind(temp01,temp01[,1]%in%wLib$STRING_id)
  colnames(temp01)<-c("string","exist")
  checkEx<-matrix(temp01$exist,ncol=1)
  row.names(checkEx)<-temp01$string

  print("Matrix construction")
  pb<-txtProgressBar(min=0,max=nrow(wLib),initial=0,style=3)
  for(i in 1:ncol(netMat)){
   if(any(is.na(checkLs[[i]])))next
   temp<-checkLs[[i]]
   temp<-temp[checkEx[temp,1]]
   if(length(temp)==0)next
   netMat[paste0("X",temp),i]=1
   setTxtProgressBar(pb,i)
  }
  close(pb)
}
#-------------------------------------------------------------------------------------------------
# Network propagation scoring - markov random walk
#-------------------------------------------------------------------------------------------------
{
  message("[Network propagetion]")
  rownames(wLib)<-paste0(wLib$ENTREZ,"|",wLib$STRING_id)
  wLib$rHit<-0
  remHit<-logical(length=nrow(mGs))
  for(i in 1:length(remGroup)){
   remHit<-remHit|(mGs$contain==remGroup[i])
  }
  mGs$rHit<-remHit
  wLib[rownames(mGs)[mGs$rHit],]$rHit<-1
  remRWinit<-as.integer(wLib$rHit)
  remRWres<-random.walk(remRWinit,netMat,r=remRWr,niter=iteration,thresh=rThresh)
  wLib$remRWres<-remRWres$p.inf[,1]
  gc(verbose=F)
  if(avoidBool){
   aRWinit<-as.integer(wLib$aScore)
```

```
  aRWres<-random.walk(aRWinit,netMat,r=aRWr,niter=iteration,thresh=rThresh)
  wLib$aRWres<-aRWres$p.inf[,1]
  rm(aRWres)
 }
 gc(verbose=F)
 if(searchBool){
  sRWinit<-as.integer(wLib$sScore)
  sRWres<-random.walk(sRWinit,netMat,r=sRWr,niter=iteration,thresh=rThresh)
  wLib$sRWres<-sRWres$p.inf[,1]
  rm(sRWres)
 }
 gc(verbose=F)
 filtLib<-wLib[rownames(wLib)%in%rownames(mGs),]
 mGs$aRWres=rep(0,nrow(mGs))
 mGs$sRWres=rep(0,nrow(mGs))
 mGs$remRWres=rep(0,nrow(mGs))
 for(i in 1:nrow(filtLib)){
  mGs[rownames(filtLib)[i],"remRWres"]=filtLib$remRWres[i]
  mGs[rownames(filtLib)[i],"aRWres"]=filtLib$aRWres[i]
  mGs[rownames(filtLib)[i],"sRWres"]=filtLib$sRWres[i]
 }
}
#-----------------------------------------------------------------------------------------------
# cScore calculation
#-----------------------------------------------------------------------------------------------
{
 message("[Calculating scores]")
 temp<-colnames(mGs)[grepl("res$",colnames(mGs))]
 for(i in 1:length(temp)){
  mGs[,gsub("res$","Adj",temp[i])]=
   mGs[,temp[i]]/sum(mGs[,temp[i]])*
   calcWeight[[gsub("RWres$","",temp[i])]]/2*
        (-1+2*as.integer(grepl("^s",temp[i])))*(1+as.integer(grepl("^s",temp[i])))
 }
 mGs$cScore<-rowSums(mGs[,colnames(mGs)[grepl("Adj$",colnames(mGs))]])
```

```
 if(any(grepl("Score$",colnames(unmGs)))){
  unmGs$cScore<-rowSums(unmGs[,colnames(unmGs)[grepl("Score$",colnames(unmGs))]])
 }
}

#-----------------------------------------------------------------------------------------------------
# Result filtering
#-----------------------------------------------------------------------------------------------------
{
 message("[Data filtration]")
 fmGs<-mGs
 if(length(LPfilt)>0){
  for(i in 1:length(LPfilt)){
   fmGs<-fmGs[fmGs[,LPfilt[i]]>=LPfiltThrs[i],]
  }
 }
 ffmGs<-fmGs[fmGs$contain%fin%lookupGroup,]
}
#-----------------------------------------------------------------------------------------------------
# Processed Data output
#-----------------------------------------------------------------------------------------------------
{
 write.csv(dataF,"OriginalData.csv",row.names = F)
 write.csv(mGs,"mappedGenes.csv",row.names = F)
 write.csv(umGs,"unMappedGenes.csv",row.names = F)
 write.csv(fmGs,"mappedGenesFilt1.csv",row.names = F)
 write.csv(ffmGs,"mappedGenesFilt2.csv",row.names = F)
}
```