

NetDecoder Protocol for Building Context-Specific Protein Interaction Networks from Transcriptomic Data Using Information Flow Modeling

Easton Blissenbach¹, Cheng Zhang¹, Carter Caya¹, Ian D. Li¹, Zhuofei Zhang¹, Shizhen Zhu^{1,2}, Hu Li^{1,3,4}, Choong Yong Ung¹, Cristina Correia¹

¹Department of Molecular Pharmacology and Experimental Therapeutics, Mayo Clinic College of Medicine and Science ²Department of Biochemistry and Molecular Biology, Mayo Clinic College of Medicine and Science ³Division of Gastroenterology and Hepatology, Department of Medicine, Mayo Clinic ⁴Paul F. Glenn Center for Biology of Aging Research, Mayo Clinic

✉ Corresponding Author : Cristina Correia <correia.cristina@mayo.edu>

Citation

Blissenbach, E., Zhang, C., Caya, C., Li, I.D., Zhang, Z., Zhu, S., Li, H., Ung, C.Y., Correia, C. NetDecoder Protocol for Building Context-Specific Protein Interaction Networks from Transcriptomic Data Using Information Flow Modeling. *J. Vis. Exp.* (), e70869, doi:10.3791/70869. (2026).

Abstract

Differential expression analysis is a commonly used technique for determining potential therapeutic targets, but it overlooks the complexity of gene networks across biological pathways. Often, highly expressed genes do not necessarily explain the properties of a biological phenotype. NetDecoder, a network biology tool that integrates transcriptomic data with protein-protein interaction (PPI) networks to model context-specific information flow, gene utility, and key edges and differentially utilized gene networks, was developed to address this limitation of differential expression analysis.

This protocol is a beginner-friendly, step-by-step guide to using NetDecoder, with comprehensive guidelines spanning data preprocessing, NetDecoder execution, and analysis of outputs. The workflow includes software configuration, network construction, and flow-based modeling analysis to quantify gene (node) and gene-gene interaction (edge-level) differences between biological conditions. The resulting outputs include key targets and routers, differential flow subnetworks, and edge flow distributions, enabling identification of key regulatory genes and pathways associated with specific biological states. After following the outlined steps, researchers will be able to conduct independent research to uncover phenotypic gene flow across phenotypes using transcriptomic data and curated PPI networks.

Introduction

The selection of genes and associated pathways for therapeutic investigation is often driven by differential expression analyses¹. This analysis method is effective at determining how gene expression differs between two or more conditions. However, genes function within complex, interconnected biological networks, meaning that individual gene expression does not

fully capture how genes interact in a network and their gene-gene relationships². Network propagation methods integrate gene expression with interaction networks to identify biologically important genes that might be missed by differential expression alone³. Current approaches do not explicitly quantify the functional importance of genes and how biological information is redistributed within protein-protein interaction (PPI) networks across biological conditions. Therefore, a method for modeling

condition-specific network behavior and quantifying changes in information flow across biological systems was needed. To account for how genes interact within a broader biological network, NetDecoder, a network biology platform, was developed to uncover genes with the greatest information flow difference between conditions. NetDecoder uses a process-guided flow algorithm to translate existing knowledge of the human PPI network, in combination with bulk RNA sequencing data, to construct a model of information flow-driven interactions⁴.

Using information flow data for phenotypic networks, a gene utility model (GUM)⁵ can be developed to identify genes with high information flow as having the highest overall gene utility within a network, regardless of their differential expression values. This approach supports more effective target prioritization strategies and identification, providing insights that traditional analysis often misses. Unlike traditional differential expression or correlation-based network methods, NetDecoder quantifies both gene (node-level) and interaction (edge-level) changes in information flow, enabling the identification of functionally important genes, even in the absence of large expression changes^{4,5}. NetDecoder is widely applicable to bulk RNA sequencing datasets that involve comparative analysis between two biological conditions, enabling the identification of changes in information flow and network organization. Although NetDecoder supports integration of other omics datasets, including proteomics and epigenomics, the present protocol specifically demonstrates the workflow using transcriptomic data. In these applications, users can define source genes based on proteins or epigenetically regulated genes, allowing information flow analysis to be initiated from these molecular features. This flexibility enables the incorporation of multi-omics evidence into network-based analyses and facilitates the discovery of cross-modal regulatory mechanisms underlying phenotypic differences.

Common study designs involve binary comparisons, including but not limited to disease versus healthy conditions, treatment responders vs non-responders, drug treatments, knockdowns or knockouts versus control experiments, and analyses of developmental or cellular state transitions. The goal of this protocol is to illustrate a reproducible framework for applying NetDecoder to uncover genes with altered network influence across biological conditions. This is achieved through easy-to-follow steps for setting up and running NetDecoder, as well as examples to follow along with basic troubleshooting techniques, and methods for result interpretation are also laid out in the protocol.

Protocol

This study used publicly available RNA sequencing datasets and did not directly involve human or animal subjects. Therefore,

institutional review board approval and informed consent were not required.

NOTE: NetDecoder requires the following input files: normalized gene expression across two defined biological conditions; a metadata file describing the biological conditions; a list of source genes for network building and modeling; a protein-protein interaction (PPI) network from the public domain; and an edge-weighted network (EWN) built for each biological condition.

1. Data preparation

1. Retrieve bulk RNA sequencing expression data and metadata

1. Download RNA sequencing data (matrix of raw gene expression counts and sample names) and corresponding metadata (sample names, conditions, etc.) from a public repository, such as the Gene Expression Omnibus (GEO), or use lab-generated experimental datasets. Typically, this is a matrix in which rows correspond to gene names and columns correspond to sample names.

NOTE: The example dataset used to demonstrate this protocol was obtained from the National Omics Data Encyclopedia (NODE) repository (BioSino database) under accession OEP001105⁶. Users may substitute their own bulk RNA sequencing datasets.

2. Generate a sample annotation file (metadata) by including sample names and their groups or associated conditions; for example, “control” and “disease” are common condition types. Confirm that the sample names match between the metadata and expression files.

2. Aggregate expression data and metadata

1. Ensure that the generated data matrix includes expression counts, gene names, and a metadata file with sample names and conditions. If necessary, use an R package such as `org.Hs.eg.db` (human) or `AnnotationDbi` to match different gene identifiers.
2. Trim out non-essential information from the metadata file using R or a similar programming language.
3. Combine the information from the expression count files into a single file for easy manipulation.

3. Filter expression data

1. Pre-process the gene expression data matrix by excluding gene duplicates, null values (NA), low expression (<10 total counts), and/or low variance genes, etc.

4. Data pre-processing

1. RNA sequencing normalization and differential expression analysis

NOTE: This step is one (1.4.1) of two (1.4.2) approaches for selecting genes for Edge Weighted Network (EWN) creation and source gene selection (steps 1.5–1.6) when beginning the analysis from bulk RNA sequencing data. As an alternative, Pearson correlation-based template matching may be utilized by skipping ahead to step 1.4.3.

1. Perform Gene ID conversions at this stage using the R package, AnnotationDbi, in concurrence with individual organism packages. NetDecoder uses gene symbols (i.e., Gene ID) that match the example PPI.
2. Perform normalization and differential expression analysis between two conditions using the R packages limma, edgeR, DESeq2, or comparable tools.
3. If using DESeq2, construct a DESeq dataset using the DESeqDataSetFromMatrix() function with a count matrix (genes and samples) and sample metadata (conditions) as input.
4. Use the DESeq() function with default settings on the object created in step 1.4.1.3 to compute differential expression values.
5. Save the results in a data matrix with gene identifiers (Gene ID) as row names and key outputs including log2 fold change (log2FC), p-value, and adjusted p-value as columns.
6. Optionally, filter the results by adjusted p-value (<0.05) and/or log2FC values (for example: |log2FC| > 1) using dplyr or a comparable tool.
7. Ensure that all processed gene lists and log2 fold change matrices are exported as tab-delimited (.txt or .csv) files for input into NetDecoder.
8. Perform pairwise comparisons for step 1.4.1 if more than 2 biological conditions are used and obtain pairwise results using the results() function from the DESeq object generated in step 1.4.1.5.

2. Microarray expression normalization - Optional

NOTE: If using microarray data, perform this step for data normalization.

1. Normalize data by opening the NetDecoder normalization script (HuLiLab/NetDecoder_Example/raw_data/NetDecoder_normalize.R), accessible from https://github.com/HuLiLab/NetDecoder_Example/tree/main, in an R-development platform and edit the

italicized portion to match the working directory containing cell intensity files (CEL).

```
setwd(~/.NetDecoder_Example/raw_data/  
CEL_files)
```

3. Template matching - Optional

NOTE: This is the second optional approach if differential expression analysis (as laid out in steps 1.1–1.4.1 of “Data Preparation”) is not conducted.

1. For RNA sequencing data not processed through limma, edgeR, and DESeq2, perform normalization externally before applying template matching. For microarray data, use the normalized expression values generated in steps 1.1–1.4.2 directly.
2. Choose the control condition as the baseline template for gene expression and compare normalized gene expression values from all other conditions of interest.
3. Calculate Pearson correlation coefficients between each gene's expression profile and the selected template. It is recommended to retain genes exhibiting statistically significant correlations ($p < 0.05$) and an absolute correlation coefficient above a user-defined threshold (such as $|r| > 0.7$).

5. Selection of source genes

NOTE: The input for this step consists of the normalized gene expression matrix developed in steps 1.1–1.4.

1. Choose a set of significant genes to be used as source genes. For differential expression-based workflows, choose adjusted p-value and log2FC thresholds (such as $\text{adj } p\text{-value} < 0.05$ and $|\log_2\text{FC}| > 2$). For template-matching workflows, select source genes from the significantly correlated genes identified in step 1.4.3.3 using the chosen correlation and significance thresholds. Thresholds should be chosen to yield approximately 300–1,000 genes for downstream network construction.

NOTE: Source genes are where the information flow starts and propagates through the network.

6. Build an edge weighted network (EWN) for each phenotype

NOTE: The following inputs are required for this step: the normalized gene expression matrix developed in steps 1.1–1.4; a protein-protein interaction (PPI) network formatted as an edge list (geneA-geneB pairs); a sample annotation file (metadata) specifying biological condition labels for each sample. The PPI network is shared as an R object and was constructed based on the iRefIndex protein database. The version used in this protocol includes all direct interactions,

but self-loops and multiple edges were excluded. The PPI network contains 15,608 proteins and 180,044 interactions. For additional details, see the NetDecoder-Methods section⁴.

1. Open the NetDecoder Edge Weighted Network (EWN) script (HuLiLab/NetDecoder_Example/input/NetDecoder_Create_EWN.R, accessible from https://github.com/HuLiLab/NetDecoder_Example/tree/main)⁴, in an R-development platform.
2. Edit the portions of the script, indicated by comments (and italicized below), to set user-specific paths and input files for EWN development.


```
path<~/NetDecoder_Example/input/
This is the path to the working directory
expQuery <- get(load("expBreastCancer_15Set2014.R"))
This is an example of a normalized expression matrix Rdata object
stQuery < read.csv("stBreastCancer.csv")
This is the sample metadata
```
3. Filter the expression matrix to retain only genes present in the PPI network.
4. For each condition, subset samples by phenotype and compute pairwise gene-gene correlations across all edges defined in the PPI network using Pearson correlation. Calculate the correlations using the processed, normalized expression matrix generated in step 1.4.
5. For each gene pair, calculate the correlation coefficient, absolute correlation, and associated p-value.
6. Remove incomplete cases (e.g., NA values).
7. Export one edge-weighted network per condition as a tab-delimited file (no header) with the following columns: proteinA, proteinB, abs_cor, cor, and pvalue.
8. Execute the R script to create co-expression networks for all relevant conditions.

2. Installing and Configuring NetDecoder

1. Download NetDecoder
 1. Access the NetDecoder software at (<https://netdecoder.hulilab.org/#ver>), choosing R or Java.
2. Install required software
 1. Install Oracle JDK and R for working/analysis environments.
 2. Install required R packages as listed in the NetDecoder documentation, using Bioconductor (<https://netdecoder.hulilab.org/#ver>)⁴.

3. Download dependencies

1. Download the necessary dependencies for NetDecoder, which include gene ontology data (accessible from <https://geneontology.org/docs/download-ontology/>), as well as the gene association reference guide (accessible from https://www.ebi.ac.uk/GOA/human_release).

4. Set up the working directory

1. Move all downloaded files (expression data, generated co-expression networks for each condition, gene ontology data, and association reference guide) into one folder, which will be the NetDecoder working directory.

5. Add the necessary folder

1. Download the NetDecoder folder (https://netdecoder.hulilab.org/wp-content/uploads/2025/07/NetDecoder_Example.zip) to the NetDecoder working directory.
2. Unzip the NetDecoder folder.

3. Running NetDecoder

1. Open the analysis bash script

1. Open a terminal running GNU Bash and navigate to the NetDecoder working directory.
2. Run the command `nano NetDecoder_Analysis.sh` to open the bash script.

2. Modify the scripts

NOTE: Most of the code is already written. In this step, the required parameters are specified in the script at the indicated locations.

1. Make a short name:

Edit the italicized portion to be the conditions being compared: `myshortname='Your_shortname_here'`

2. Set software paths:

Edit the following three path definitions to the correct paths by editing the italicized portions:

`JAVA="/your/path/here"`

`export R="/your/path/here"`

`alias R="/your/path/here"`

3. Set working directory:

Edit the italicized portion to be the working directory where all files will be accessed from: `INPUT_DIR="/your/path/here"`

4. Set NetDecoder library:

Edit the italicized portion to be the path to the NetDecoder library installed in step 2.4 from the zip file:
LIB_DIR="/your/path/here/netdecoder_lib"

5. Set gene ontology and association paths:

Edit the following italicized portions of the two directories to the locations of the preferred gene ontology/association files.

SYMBOL=\$LIB_DIR/gene_association_file

GO=\$INPUT_DIR/gene_ontology_file

6. Set input arguments:

Update the italicized portions of the following five lines according to the data, as per the commented instructions found in the reference NetDecoder script.

geneList=\$INPUT_DIR/gene_file

state_trt=Treated Condition

state_ref=Reference Condition

PPI_trt=

\$INPUT_DIR/treated_co_expression_network_file

PPI_ref=\$INPUT_DIR/

reference_co_expression_network_file

7. Set the proper stage:

Uncomment (by removing the # symbol) the desired stage of execution, and re-comment it once the stage is completed. Stage one is gen_net_trt, followed by gen_net_ref, then the analysis stage, and finally the collect stage.

Set the desired stage to execute, beginning with stage one.

#STAGE="gen_net_trt"

#STAGE="gen_net_ref"

#STAGE="analysis"

#STAGE="collect"

8. Navigate to the main Bash terminal:

Press Ctrl+X, then Y, to save and exit the bash script.

3. Execute NetDecoder

NOTE: If it runs successfully, logging messages appear in the terminal.

1. Navigate to the terminal and run ./NetDecoder_Analysis.sh.

2. Execute the stages in the order listed in step 3.2.7 until all four have been completed individually.

4. Retrieve results

1. Navigate to the output files and graphical summaries.

Results

As shown in **Figure 1**, NetDecoder operates through a structured workflow consisting of data processing, network configuration, and flow-based analysis, with outputs organized into distinct directories corresponding to each stage of the pipeline. This modular structure enables systematic validation of outputs, ensuring that results can be traced back to each computational step and supporting overall reproducibility.

Successful execution of NetDecoder (**Figure 2**) produces output across four directories (*analysis*, *collect*, *networks*, and *your_shortname*) containing data corresponding to various conditions, including figures and quantitative outputs for flow values, subnetworks, and other flow-interaction outputs (**Figure 1**, **Figure 2**, and **Figure 3**). The presence of populated files and figures across these directories indicates that the pipeline executed properly. In contrast, failed runs are characterized by missing outputs and/or incomplete figures. Because NetDecoder evaluates thousands of genes within a PPI network, successful runs typically require multiple hours of computation, depending on the dataset size and available resources. For the representative human biliary tract cancer analysis presented here, execution of NetDecoder required a runtime of approximately 4–6 h on the Linux-based Puget3 (Puget Systems) workstation. Unusually short runtimes may indicate improperly formatted input data or incomplete pipeline execution. Improperly formatted input data or failed pipeline execution may result in error messages that can be traced back to the cause of the issue. **Figure 3** illustrates representative outputs that are present after successful NetDecoder execution. The representative analysis presented here utilized RNA sequencing data from 255 biliary tract cancer samples (OEP001105)⁶, enabling the comparison of stage I–II and stage III–IV disease states and the altered information flow between conditions.

The three heatmaps in **Figure 4** summarize the overall changes in network information flow between genes across conditions, including router genes that transmit substantial information flow within the network, important downstream target genes, or highly impacted overall genes. A broad distribution of both positive and negative differential flow indicates that information is redistributed across the network, rather than uniformly increased or decreased. This heterogeneity and variety of gene types support NetDecoder's ability to identify genes with altered functional importance between phenotypes.

Figure 4 also shows edge-level bar plots that quantify flow changes for individual gene-gene pair interactions across conditions. These results demonstrate that flow through specific interactions can vary across conditions, reflecting context-dependent network rewiring. Thus, edge flow captures interaction-level changes in

information transfer, whereas differential flow provides a node-level summary of these changes, enabling prioritization of genes with the greatest overall shift in network influence.

Together, these outputs demonstrate that NetDecoder captures gene (node-level), gene-gene (edge-level), and network-level alterations in information flow within biological networks (Figure 4). The heatmaps identify genes with altered information propagation, routing, and reception in the network, whereas edge-level analyses reveal the specific interactions driving these changes. The phenotype-specific information networks provide a visual representation of network rewiring between conditions, where nodes represent genes and edge thickness corresponds to information flow magnitude (Figure 4). This systems-level representation supports the identification of key regulatory genes and pathways associated with the conditions being studied. Furthermore, the successful production of these figures indicates that NetDecoder has been executed correctly.

Because NetDecoder produces a large number of output files, identifying the most relevant results is essential for interpretation. For example, to examine flow differences between conditions (e.g.,

low stage vs. high stage), two key files can be used (navigation steps are guided by the file structure of Figure 3). The first, "EDGE_CENTERED_SUBNET_flowDifference_Disease.txt" (located in the analysis/Disease directory), identifies interactions (edges) with the greatest changes in flow between conditions. The second, "flowDifference_PRIORITIZED_NETWORK.txt" (located in the "your_shortcode" directory), identifies genes (nodes) with the greatest flow differences. Together, these files provide a comprehensive view of both interaction-level and gene-level changes in network behavior.

More broadly, the "analysis" folder contains information on network routers, key targets, and important differential flow genes. The "your_shortcode" folder contains raw text files with phenotype-specific data, such as total flow values and key targets and routers. The networks folder contains the subnetworks generated by NetDecoder, which can be further analyzed and visualized in Cytoscape⁷. Finally, the "collect" folder contains consolidated data and figures that provide a general summary of the results. Further visualization and analysis of NetDecoder results can be performed using R packages such as ggplot2, igraph, pheatmap, etc.

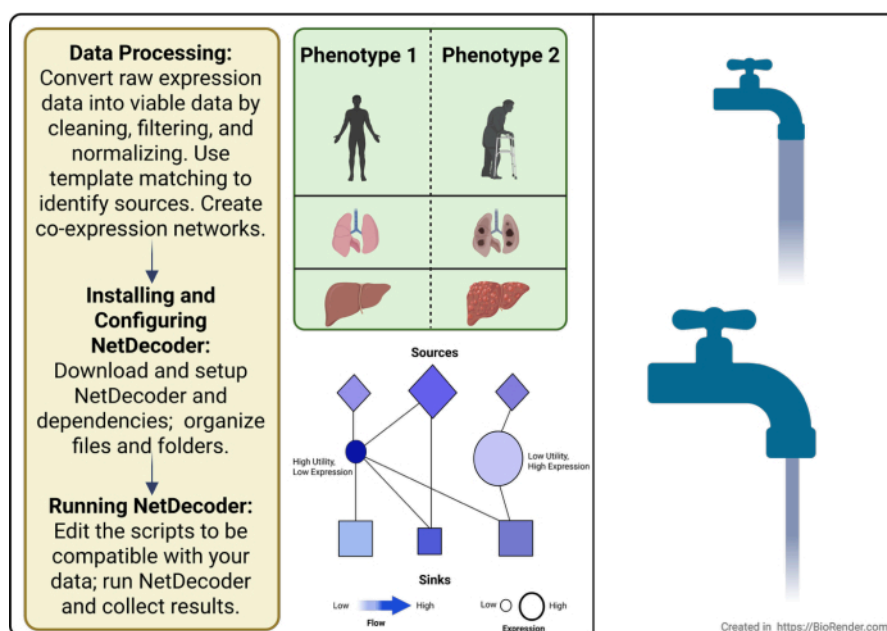


Figure 1: Representation of NetDecoder Pipeline and General Functionality. NetDecoder requires three general stages (yellow box): data processing, NetDecoder configuration, and NetDecoder execution. Each step of these stages must be closely followed to ensure successful results. NetDecoder requires expression data from two conditions (examples in green box) to predict which genes may be associated with high information flow and gene utility. In the water faucet analogy, gene expression levels are illustrated by the size of the faucet, i.e., how wide it can open, while the gene utility or activity reflects the actual water flow coming through the faucet, illustrating how much this route is functionally being used. As shown in the figure, a highly expressed gene (large faucet size) can have low gene utility (low level of water flow) while a gene with a low level of expression (small faucet size) can have high gene utility (high levels of water flow). This illustrates how genes may be of higher overall significance in a given condition, even if expression levels are lower than those of another gene. The illustration in the bottom center shows how genes can be interconnected and experience different levels of flow (color), regardless of how much they are expressed (size). Created in BioRender. Blissenbach, E. (2026) <https://BioRender.com/8okynbu>. [Please click here to view a larger version of this figure.](#)

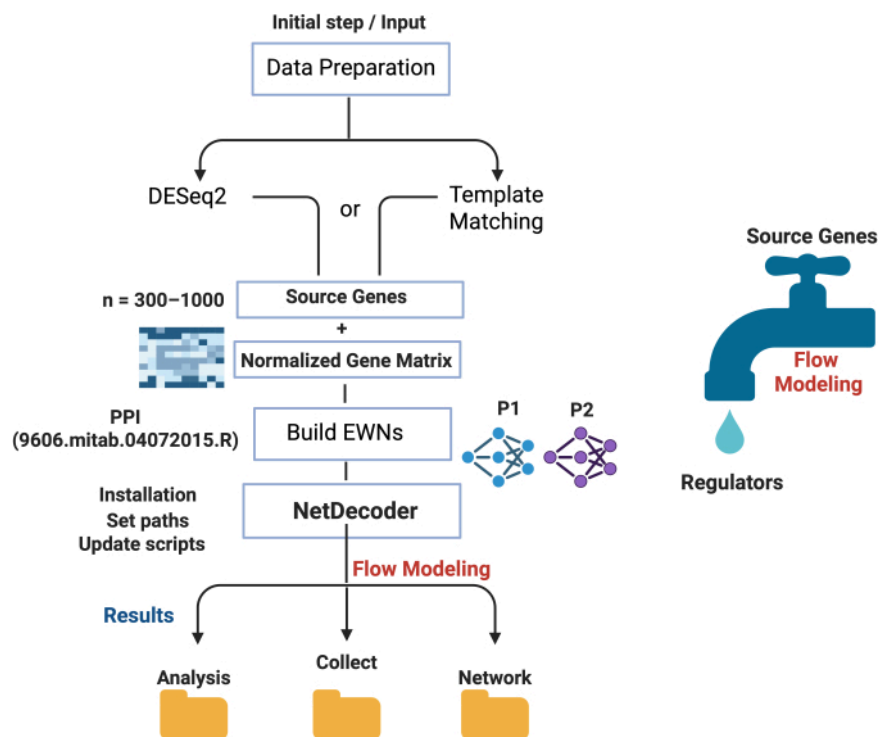


Figure 2: NetDecoder General Workflow. The core principle of the NetDecoder algorithm is to model gene utility in a protein-protein interaction (PPI) network through information flow analysis. The workflow begins with data pre-processing and edge-weighted network (EWN) building. Gene expression data from two biological conditions, phenotype 1 (P1) and phenotype 2 (P2) are processed to identify source genes using either differential expression or template matching approaches. Normalized expression matrices are then used to build condition-specific EWNs, where edge weights reflect gene-pair relationships in each phenotype. NetDecoder subsequently quantifies differences in information flow between conditions, enabling the identification of network rewiring events and alterations in gene utility. Outputs include differential information flow scores, context-specific information networks, impact score heatmaps, and additional network-level and gene-level metrics that facilitate downstream biological interpretation and mechanistic discovery. Created in BioRender. Correia, C. (2026) <https://BioRender.com/29cmswf>. [Please click here to view a larger version of this figure.](#)

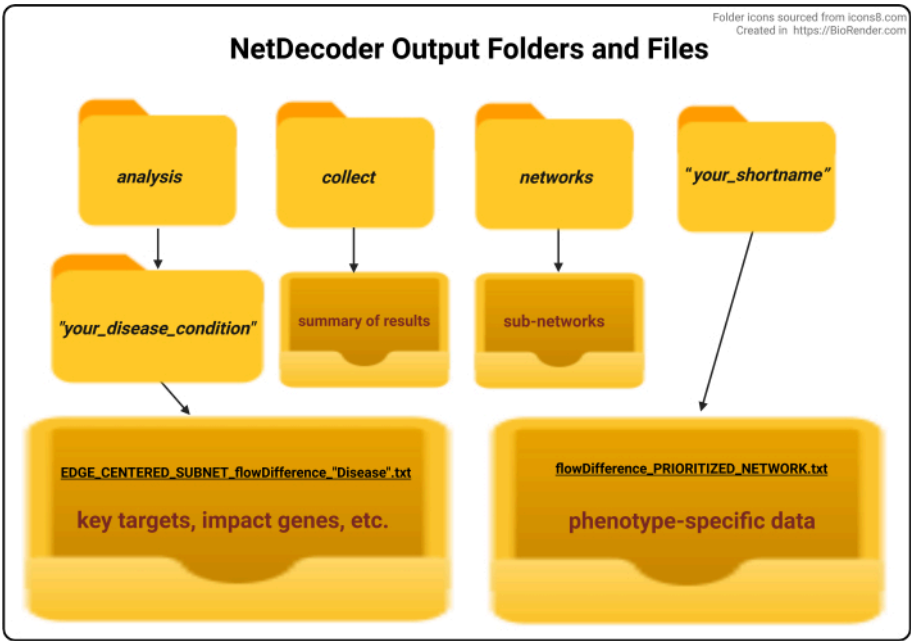


Figure 3: NetDecoder Outputs and Folder Structure. This diagram represents the output directory structure used to access result files for interpretation. Represented in italics are folder names (on the folder icons), with those in quotations being specific to analysis nomenclature. Values on inbox icons indicate either files or file types, depending on the color coding. Red denotes the file types that can be found in each folder, while key files are shown in bold and underlined (flowDifference_PRIORITIZED_NETWORK.txt and EDGE_CENTERED_SUBNET_flowDifference_Disease.txt). Folder icons were sourced from icons8 (<https://icons8.com>). Created in BioRender. Blissenbach, E. (2026) <https://BioRender.com/8okynbu>. [Please click here to view a larger version of this figure.](#)

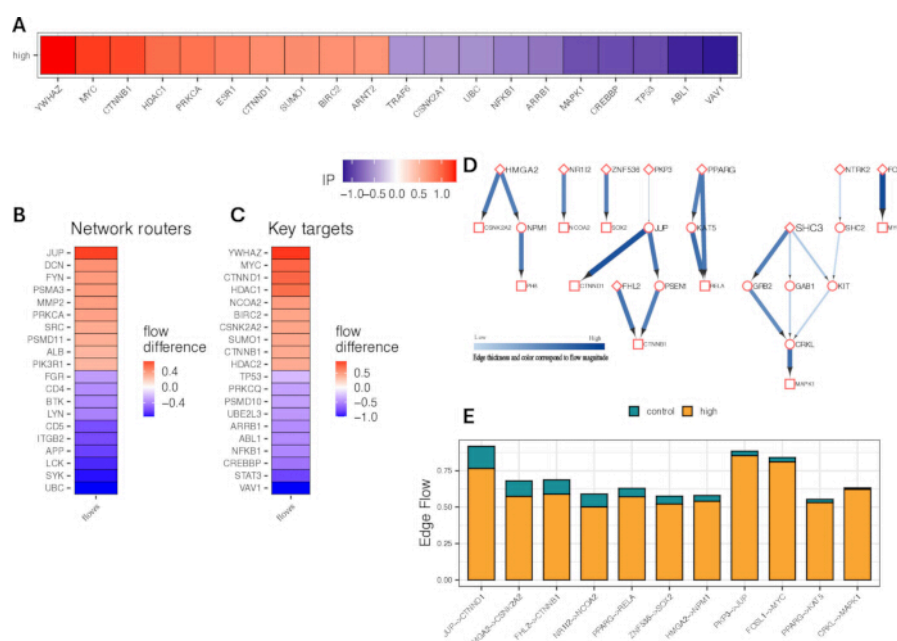


Figure 4: Example of Results Generated by NetDecoder. Impact genes (A), network routers (B), and key targets (C) heatmaps, as well as context-specific networks (D) and edge flow bar graphs (E), are key outputs from NetDecoder. In this example, a biliary tract cancer expression dataset (OEP001105) was used to compare patients with early-stage disease (stages I-II, low) and advanced-stage disease (stages III-IV, high), and NetDecoder was applied to identify genes with high differential information between the two groups. Each plot shows red indicating increased flow and blue indicating decreased flow. Network routers (B) are key intermediary genes where large amounts of flow pass through (collect/Disease_Network_routers.pdf), key targets (C) are important downstream regulators (collect/Disease_Key_targets.pdf), and the flow difference heatmap represents the overall change in gene-level information flow between conditions (analysis/flowDifference_heatmap.pdf). A phenotype-specific information network (D) can be visualized, with each gene representing a node and gene-gene interactions represented as edges (lines) (analysis/EDGE_CENTERED_SUBNET_Disease). Edge thickness corresponds to the magnitude of information flow between genes. The bar graph (E) shows differences in edge flow across gene-gene interactions between the two selected phenotypes, with low stage pairs represented in teal and high stage pairs in orange (analysis/Disease_keyEdges.pdf). [Please click here to view a larger version of this figure.](#)

Discussion

This protocol outlines the implementation of NetDecoder, a network-biology framework that integrates gene expression data with protein-protein interaction (PPI) networks to model condition-specific information flow and prioritize genes based on their functional influence within biological systems. The successful application of this method depends on several critical steps, careful methodological choices, and proper interpretation of outputs.

The initial stages of the protocol include retrieving expression data, generating metadata, and constructing a unified expression matrix. These steps are critical for ensuring compatibility with downstream analysis. The expression matrix must be formatted with genes as rows and samples as columns, with sample names in the matrix and metadata file matching identically. Any inconsistencies at this stage (e.g., mismatched sample identifiers or duplicated gene names) will propagate through the pipeline and result in failures in later steps.

Filtering low-quality data (e.g., genes with low counts, missing values, or low variance genes) is particularly important, as these features can introduce noise into correlation-based network construction. A successful preprocessing step is indicated by a clean expression matrix with no missing values and consistent gene identifiers that match those used in the PPI network.

A key decision point in the protocol is the choice between differential expression analysis and template matching for selecting source genes used in Edge Weighted Network (EWN) construction. Differential expression analysis is most appropriate when comparing well-defined experimental groups with sufficient replicates. This approach identifies genes with statistically significant expression changes between conditions and provides quantitative outputs such as log2 fold change (log2FC) and adjusted p-values. Successful execution is indicated by a distribution of both significant and non-significant genes, rather than uniformly null results. In contrast, template matching is better suited when the goal is to identify genes with expression patterns correlated with a continuous or reference profile. This method retains genes based on correlation strength rather than the magnitude of differential expression. A successful

template-matching step yields a set of genes with statistically significant correlations with the reference condition. The choice between these approaches affects downstream network topology; differential expression emphasizes magnitude changes, whereas template matching emphasizes coordinated expression patterns.

The EWN construction step is one of the most critical components of the protocol. Here, normalized gene expression data are integrated with a PPI network to compute pairwise gene-gene correlations for all interactions present in the network. Only genes shared between the expression dataset and the PPI network are retained, ensuring biological relevance and computational consistency. For each condition, Pearson correlation coefficients are calculated across all edges, along with corresponding p-values and absolute correlation values. These metrics define the edge weights used by NetDecoder. Successful EWN construction is indicated by thousands of gene-gene correlations, broad distribution of correlation values, and minimal missing values after filtering. Failure at this stage is often due to mismatched gene identifiers, insufficient sample size, or improperly normalized expression data. Common NetDecoder failure points can often be resolved through methodical verification of input files and software configuration. If NetDecoder terminates unexpectedly or produces incomplete outputs, users will need to check that sample identifiers are identical between expression matrices and metadata files, that gene identifiers are consistent across expression datasets, source gene lists, and PPI network, and that expression matrices contain no missing values. If errors or truncated results occur, users can verify software installation and version compatibilities, inspect log messages generated during execution, and confirm successful completion of each intermediate step before proceeding to downstream analyses. Inspection of the output files and/or console messages can help localize the source of an error before subsequent analyses are attempted. NetDecoder requires meticulous configuration of file paths, input arguments, and dependencies. Key steps include specifying the working directory and library paths, gene ontology and annotation files, condition-specific EWN inputs, and source gene lists.

Because the pipeline is executed in stages (network generation of both conditions, analysis, and result collection), errors in earlier stages will prevent successful completion of subsequent ones. A properly functioning run produces four directories (*analysis*, *collect*, *networks*, *your_shortname*), each containing condition-specific results. Another practical indicator of correct execution is runtime. Successful runs typically require several hours to process because NetDecoder evaluates large interaction networks; extremely short runtimes often indicate misconfigured inputs or skipped computational steps.

NetDecoder differs fundamentally from traditional gene prioritization and network analysis approaches. Methods such as

Weighted Gene Co-expression Network Analysis (WGCNA)⁸ cluster genes based on correlation structure, but do not incorporate directional flow or quantify how information propagates through a network. Similarly, pathway enrichment analyses⁹ identify functional overrepresentation, but do not account for interaction-level dynamics or changes in network connectivity.

NetDecoder uses an integrated approach by combining gene expression data with PPI networks to model condition-specific information flow. By quantifying both individual gene (node-level) scores and interaction (edge-level) flow changes, it captures how network structure and information routing are rewired between conditions. This enables identification of genes that may not exhibit strong differential expression while still playing a central role in mediating network behavior, consistent with the gene utility model (GUM)⁵, which posits that genes with high differential information flow drive condition-specific network function.

Despite its strengths, NetDecoder has several limitations. First, it is a computational framework that infers gene importance based on modeled network information flow rather than direct experimental evidence¹⁰. As such, its predictions should be interpreted as hypotheses that require validation through biological experiments. Functional validation approaches, such as gene knockout studies¹¹ or targeted perturbation assays¹², are essential to confirm whether genes identified as high utility by NetDecoder truly influence the biological processes or disease states being studied. Second, the method is highly dependent on the quality and completeness of the underlying PPI network. Because PPI databases are often biased toward well-studied genes, less characterized interactions may be underrepresented, potentially limiting the discovery of novel regulatory relationships. Variability in sample size, experimental design, and data quality can also influence network construction and downstream flow calculations. Third, NetDecoder does not assume that static edges fully capture dynamic information flow. Instead, NetDecoder infers context-specific activity and quantifies information flow using a PPI network as a structural prior, which serves as a scaffold to define the space of biologically plausible interactions. Dynamic behavior is then introduced by overlaying state-specific molecular data (e.g., gene expression), effectively reweighting or activating subsets of the network in a context-dependent manner.

NetDecoder's primary objective is to model information propagation over a static PPI scaffold while preserving quantitative and continuous relationships between genes (nodes). In contrast, approaches such as Boolean networks offer a simplified and interpretable representation of dynamics by modeling protein activity as discrete on/off states governed by logical interactions¹³. These approaches have been widely used to study gene regulatory and signaling networks under varying conditions^{14,15,16}. However, they typically require predefined logical rules and discretization of protein states, which can be challenging to define at scale for large,

heterogeneous biological PPI networks¹⁷. In this context, Boolean network modeling represents a complementary direction to NetDecoder. While NetDecoder captures continuous, quantitative information flow, integrating logical rule-based dynamics or hybrid discrete-continuous models could enhance the interpretability of condition-specific signaling behaviors. Developing Boolean-style information flow algorithms, therefore, represents a promising avenue for future work.

Future directions for NetDecoder include integrating with additional omic data types and expanding to single-cell transcriptomics to improve resolution and biological context. For example, deep learning-based approaches for spatial transcriptomic expression prediction and imputation aim to improve data quality and signal recovery, but do not explicitly model information flow across interaction networks^{18,19}. Incorporating multi-omic levels of analysis could both further enhance its predictive power and lead to more reliable results. With continued methodological development, NetDecoder will be able to produce multiple layers of information flow, providing research scientists with multi-omic validation for their data of interest.

Disclosures

The authors have no financial competing interests.

Illustrations for Figures 1 and 2 were created with BioRender (BioRender.com).

Acknowledgments

This work was supported by grants from the Mayo Clinic Center for Biomedical Discovery, the Mayo Clinic Comprehensive Cancer Center (NIH; P30 CA015083), the Mayo Clinic Center for Cell Signaling in Gastroenterology (NIH: P30DK084567), the Glenn Foundation for Medical Research, the V Foundation for Cancer Research (S.Z.), the Mayo Clinic Nutrition Obesity Research Program, the David F. and Margaret T. Grohne Cancer Immunology and Immunotherapy Program, Schmidt Sciences, and Innovation and the National Institutes of Health (NIH; U19AG74879, P50CA136393, R01CA240323, R03OD038392).

References

- Rosati D et al. Differential gene expression analysis pipelines and bioinformatic tools for the identification of specific biomarkers: A review. *Comput Struct Biotechnol J*. 2024;23:1154–68.
- Barabási A-L, Oltvai ZN. Network biology: understanding the cell's functional organization. *Nat Rev Genet*. 2004;5(2):101–13.
- Cowen L, Ideker T, Raphael BJ, Sharan R. Network propagation: a universal amplifier of genetic associations. *Nat Rev Genet*. 2017;18(9):551–62.
- Rocha DLE, Ung YC, McGehee DC, Correia C, Li H. NetDecoder: a network biology platform that decodes context-specific biological networks and gene activities. *Nucleic Acids Res*. 2016;44(10):e100.
- Ung YC et al. Gene utility recapitulates chromosomal aberrancies in advanced stage neuroblastoma. *Comput Struct Biotechnol J*. 2022;20:3291–303.
- Dong L et al. Proteogenomic characterization identifies clinically relevant subgroups of intrahepatic cholangiocarcinoma. *Cancer Cell*. 2022;40(1):70–87.e15.
- Ono K et al. Cytoscape Web: bringing network biology to the browser. *Nucleic Acids Res*. 2025;53(W1):W203–W212.
- Langfelder P, Horvath S. WGCNA: an R package for weighted correlation network analysis. *BMC Bioinformatics*. 2008;9(1):559.
- Ma J, Shojaie A, Michailidis G. A comparative study of topology-based pathway enrichment analysis methods. *BMC Bioinformatics*. 2019;20(1):546.
- Ideker T, Krogan NJ. Differential network biology. *Mol Syst Biol*. 2012;8(1):565.
- Tong C et al. Review of knockout technology approaches in bacterial drug resistance research. *PeerJ*. 2023;11:e15790.
- Wei Z et al. PerturbBase: a comprehensive database for single-cell perturbation data analysis and visualization. *Nucleic Acids Res*. 2025;53(D1):D1099–D1111.
- Chevalier S et al. Data-driven inference of Boolean networks from transcriptomes to predict cellular differentiation and reprogramming. *NPJ Sys Biol Appl*. 2025;11(1):105.
- Kadelka C, Butrie T-M, Hilton E, Kinseth J, Schmidt A, Serdarevic H. A meta-analysis of Boolean network models reveals design principles of gene regulatory networks. *Sci Adv*. 2024;10(2):eadj0822.
- Klamt S, Saez-Rodriguez J, Lindquist JA, Simeoni L, Gilles ED. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*. 2006;7(1):56.
- Saez-Rodriguez J et al. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Mol Syst Biol*. 2009;5(1):331.
- Bornholdt S. Boolean network models of cellular regulation: prospects and limitations. *J R Soc Interface*. 2008;5(suppl_1):S85–S94.
- Yuan L, Jiang Y, Meng B, Wang Q, Wang C, Huang D-S. SpaLSTF: Diffusion-based generative model with BiLSTM and XCA-Transformer for spatial transcriptomics imputation. *PLoS Comput Biol*. 2026;22(2):e1013954.
- Yuan L, Jiang Y, Wang Q, Hu C, Li G, Zheng C-H. DeepSGE: predicting spatial gene expression using residual network with efficient channel attention and dynamic graph attention network. *BMC Genomics*. 2026;27(1):293.