bio-protocol

Published: Jan 05, 2023

A Cartographic Tool to Predict Disease Risk-associated Pseudo-Dynamic Networks from Tissue-specific Gene Expression

Chixiang Chen[1, 2], Biyi Shen[3], Lijun Zhang[4], Tonghui Yu[5], Ming Wang[4, *] and Rongling Wu[6, *]

[1]Division of Biostatistics and Bioinformatics, University of Maryland School of Medicine, Baltimore, MD, USA
[2]Department of Neurosurgery, University of Maryland School of Medicine, Baltimore, MD, USA
[3]Bristol Myers Squibb, Lawrenceville, NJ, USA
[4]Department of Population and Quantitative Health Sciences, School of Medicine, Case Western Reserve University, Cleveland, OH, USA
[5]School of Mathematics, Hefei University of Technology, Anhui, China
[6]Division of Biostatistics and Bioinformatics, College of Medicine, Penn State College of Medicine, Hershey, PA, USA
*For correspondence: mxw827@case.edu; ronglingwu@pennstatehealth.psu.edu

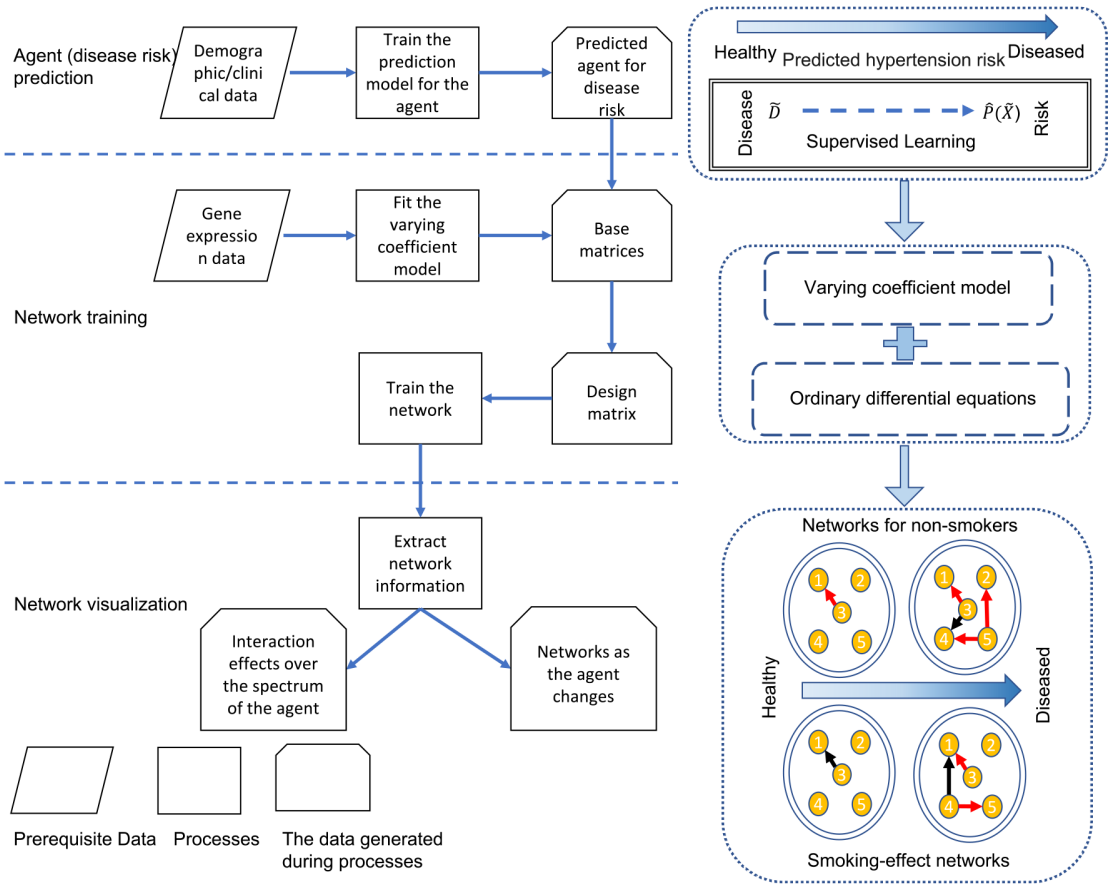
Abstract

Understanding how genes are differentially expressed across tissues is key to reveal the etiology of human diseases. Genes are never expressed in isolation, but rather co-expressed in a community; thus, they co-act through intricate but well-orchestrated networks. However, existing approaches cannot coalesce the full properties of gene–gene communication and interactions into networks. In particular, the unavailability of dynamic gene expression data might impair the application of existing network models to unleash the complexity of human diseases. To address this limitation, we developed a statistical pipeline named DRDNetPro to visualize and trace how genes dynamically interact with each other across diverse tissues, to ascertain health risk from static expression data. This protocol contains detailed tutorials designed to learn a series of networks, with the illustration example from the Genotype-Tissue Expression (GTEx) project. The proposed toolbox relies on the method developed in our published paper (Chen et al., 2022), coding all genes into bidirectional, signed, weighted, and feedback looped networks, which will provide profound genomic information enabling medical doctors to design precise medicine.

Keywords: Gene regulatory network, Genotype-Tissue expression, Quasi-dynamic, Ordinary differential equations, Statistical algorithm, Risk prediction

This protocol was validated in: Bioinformatics (2021), DOI: 10.1093/bioinformatics/btac038

Graphical abstract



Flowchart illustrating the use of DRDNetPro. The left panel contains the summarized pipeline of DRDNetPro and the right panel contains one pseudo-illustrative example. See the Equipment and Procedure sections for detailed explanations.

Background

Differential expression of genes across different tissues has been thought to play an important role in shaping human diseases (Oliva et al., 2020). An increasing body of studies has begun to characterize how genes are co-expressed in tissues, to rewire transcriptional regulatory networks as key molecular mechanisms underlying human disease (Saha et al., 2017; Malatras et al., 2020; Consortium, 2020). However, existing approaches for reconstructing gene regulatory networks are limited in capturing the full properties of gene–gene interactions, which are essential for a mechanistic understanding of disease etiology. For example, correlation-based approaches can only estimate the strength of gene–gene interactions, failing to identify the causality of the interactions, whereas Bayesian networks can identify the causality, but cannot characterize the sign of the interactions and feedback cycles. All these properties can be recovered using dynamic modeling of gene expression. However, it is impossible and ethically impermissible to collect temporal transcriptional data from human tissues. We have developed a series of quasi-dynamic models that can identify the aforementioned properties in gene–gene interactions from static data (Chen et al., 2019; Griffin et al., 2020; Wu and Jiang 2021; Chen et al., 2021). More recently, we leveraged these models to recover tissue-specific and pseudo-dynamic gene regulatory networks across the spectrum of disease risk (Chen et al., 2022). In this article, we develop a software pipeline called **D**isease **R**isk-associated pseudo-**D**ynamic **Net**works

Bio-**Pro**tocol (DRDNetPro). This pipeline provides a detailed tutorial, enabling researchers to reconstruct pseudo-dynamic networks from static gene expression data, which helps to identify context-specific and personalized networks, and further the mechanistic understanding of diseases. The software and detailed tutorials with illustrative data examples can be found in our GitHub repository (https://chencxxy28.github.io/DRDNetPro/articles/NAME-OF-VIGNETTE.html).

Equipment

Computational requirements

1. The users need to prepare a desktop/laptop computer with R version 4.1.0 or above installed.

Install packages (The summary of the tutorial in the GitHub page repository)

1. Run the function "install.packages()" to install the required packages, including "pROC", "np", "splines2", "grpreg", "Matrix", "igraph", "ggplot2", and the optional packages "ranger", "XGBoost", and "dplyr".
2. Run the function "devtools::install_github()" to install the package "DRDNetPro". Run the function "install.packages()" to install the package "devtools", if it has not been installed yet.

Software

Software list

1. DRDNetPro (Chixiang Chen, https://github.com/chencxxy28/DRDNetPro)
2. graph (The igraph core team, https://igraph.org/r/)
3. ggplot2 (Hadley Wickhan, https://ggplot2.tidyverse.org/)

Datasets for demonstration

The demo data for method illustration can be downloaded from the website https://chencxxy28.github.io/DRDNetPro/articles/web/data.html, which is from the GTEx project originally. Gene expression data were collected from blood vessels and de-identified subsamples of phenotype information from donors.

Procedure

To predict disease risk-associated pseudo-dynamic networks, we need the following inputs from each subject: predicted disease risk (named "agent" below), one covariate of interest (e.g., smoking), and pre-processed gene expression data. The detailed procedures of how to generate the agent, train the network model, and visualize the results are listed below.

All the programs, demo data for illustration, and brief tutorials for the following procedures can be found in our GitHub repository (https://chencxxy28.github.io/DRDNetPro/articles/NAME-OF-VIGNETTE.html).

A. Predict the agent (Tutorial 1 in the GitHub repository)

1. Prepare the data for training prediction values of the agent

   Before predicting the agent, users need to prepare the data for model training. The data needs to be a matrix, including the outcome in the first column and all potential predictors in the remaining columns. Do not include the intercept as one column. The outcome in this protocol has values of 0 and 1, corresponding to healthy and diseased subjects, respectively. More complicated outcomes with multiple categories are not considered in the current bio-protocol, as we regard this as future work. One remedy for the case of

multiple categories are to categorize the outcome into binaries, which is also often applied in practice. All values in the matrix should be numeric. The missing data is allowed in this training process, though it is always recommended to have complete data as an input.

For demonstration, we have prepared a toy data example in this tutorial, which can be accessed from https://chencxxy28.github.io/DRDNetPro/articles/web/data.html. See details in the Data analysis section below.

2. Select a training method

There is no unique way to predict the agent. Either statistical regression models or machine learning algorithms can be applied. In our program, we allow the users to specify the following commonly used methods: conventional logistic regression, random forest, and XGBoost.

   a. Logistic regression—This is the most conventional statistical tool to predict disease risk, which will lead to a probability-scaled risk. Run the function "glm()" in R to fit the training model, and then run "fitted()" to extract the predicted agent.

   b. Random forest—This is a machine learning method with an ensemble of decision trees. It builds and combines multiple decision trees to produce more accurate predictions. It is a non-linear classification algorithm. There are several R packages available for implementing random forest. We provide demo codes based on the package "ranger". The data required for "ranger" should be in a matrix form, where the outcome should be a factor, to enable running classification trees. After preparing the data, run the function "ranger()" to train the model, and run the function "predict()" to extract the predicted agent.

   c. XGBoost—This is another machine learning method with an efficient implementation of the gradient boosting framework. It provides built-in k-fold cross-validation (CV) Stochastic Gradient Boosting Machines, with column and row sampling (per split and per tree) for better generalization. We provide demo codes based on the package "XGBoost". A specific type of input data is required by "XGBoost", where the response and predictors should be separately stored. The series codes to prepare the required data are provided in this tutorial (see details in Tutorial 1). After preparing the data, run the function "xgb.cv()" to get an initial prediction model. Then, run the function "dplyr::summarise()" to extract the tuned parameter, i.e., the number of trees. Finally, run the function "xgboost()" to train the data, and "predict()" to extract the predicted agent.

3. Predict and evaluate the agent

   a. Prepare the training and testing data—When the agent model is fitted, it is better to check its prediction performance. Note that the pseudo-dynamic network is sensitive to the values of agent, and the agent with good prediction for the disease risk is preferred. To evaluate its predictive performance, we consider the metric of the area under the receiver operating characteristic curve (AUC-ROC). To avoid the overfitting issue, we need a testing data set. For illustration purposes, we artificially created training and testing data by randomly splitting the original data, with no overlap between the two constructed data sets. The training data is used for data fitting, whereas the testing data is used for prediction evaluation. Run the function "sample()" to generate these two datasets. In real applications, another way to validate prediction performance is to use identical but independently sampled data from an external source as the testing set.

   b. Generate the AUC-ROC plot—After fitting the model based on the training data, by running the function "glm()", extract the predicted agent based on the testing data, by running the function "predict()". Run the function "roc" to generate the plot. A larger AUC-ROC value indicates a better prediction model.

B. Train the network model (Tutorial 2 in the GitHub repository)

1. Prepare the data for training the network

   a. Filtering and transformation—To construct the gene–gene network, the users need to preprocess the gene expression data, by normalizing, excluding low-expressed genes, and, if needed, transforming to relieve skewness and reduce variance.

b. Screening—To learn the disease risk-associated network, genes showing changing expression patterns as the agent changes should be included, which is different from association-based network. Two selection procedures can be considered:
   i. To test differentially expressed genes, run the function "test_screen()" to locate genes showing the difference between disease groups, after adjusting for covariates and applying the Hochberg multiple test correction (adjusted P-value <0.05).
   ii. To screen the genes, run the function "spearman_screen()" to find genes showing high associations between expression and the predicted agent. Pick the top (e.g., top 40) most correlated genes. Perform separate screening to check if more than one homogeneous group is considered. Determine the final genes by combining the two selection procedures.
   iii. Optional screening can be done based on prior biological knowledge (i.e., the GO enrichment analysis).

2. Train the varying coefficient model
   a. Refine the data—Before constructing the network, each selected gene should be fitted by the varying coefficient model. This model estimates dynamic features of coefficients in terms of a certain agent by Kernel/Spline regression, which fits the context of reconstructing pseudo-dynamic networks well. We refer readers to Chen et al. (2022) for more technical details. When two subjects have the same agent value, it means the expression data from these two are not informative. Run the R codes in this tutorial to refine the data, by deleting the data with agents too close to each other.
   b. Fit the model—After refining the data, run the function "vc.fit()" to fit the varying coefficient model with a covariate (e.g., smoking).
   c. Check the fitted curves—Run the function "plot()" and "lines()" to check how well the data is fitted. The users can also use the package "ggplot2" for visualization.
3. Generate a base matrix for the network model
   a. The base matrix is the key component for the network to learn in a non-parametric manner. Run the function "base.construct()" to construct the base matrix for the varying intercept and base matrices for the varying coefficient of the covariate (e.g., smoking). The default knots are lower quartile and upper quartile with degree =3, which is related to the cubic spline.
   b. The output—A list of four matrices based on varying intercepts (X_big), varying covariate effects (X_big_cov), varying intercepts with a column containing ones (X_big_int), and varying covariates with a column containing ones (X_big_int_cov).
4. Train the network
   a. After constructing the base matrices, the users should run the function "network.learn()" to train the network structure. This function requires the input of the design matrix, the outcome, and the group information, as well as the method to select the tuning parameters.
   The design matrix—Extract the output results from the previous step of construction of base matrices, which includes
      i. data_observe: the gene expression matrix.
      ii. x_cov: the vector including the numerical values of the covariate of interest (e.g., smoking: 0 and 1).
      iii. X_big_int: the matrix based on varying intercepts, with a column containing ones.
      iv. X_big_int_cov: the matrix based on varying coefficients, with a column containing ones.
      v. Agent: the predicted disease risk obtained from Tutorial 1.
   b. Determine the tuning parameters—Multiple methods are available to tune the hyperparameters, including 5-fold CV (CV5), 10-fold CV (CV10), Akaike information criterion (AIC), Bayesian information criterion (BIC), and empirical BIC (EBIC). Use the argument "cv" in the function "network.learn()" to specify the preferred method.
   c. Output from the network learning—This output includes information on node size, interaction, and covariate effect. All of them will be used in the next step for visualizing the results.
      i. Node size information: self-node size for baseline, self-node size for the covariate effect, overall self-node size.

ii. Interaction information: gene interaction effects from the baseline, gene interaction effects from the covariate, overall gene interaction effects.
iii. Covariate effect information: covariate effect, trend effect for baseline, trend effect from the covariate.

C. Visualize the results (Tutorial 3 in the GitHub repository)

1. Extract the results from the trained network model
   Read the output data from the network learning in Tutorial 2, which includes network.output, data.list.t3, and gene.names.
2. Visualize the networks with a given range of agent values
   a. Baseline networks—To visualize the pseudo-dynamic network given an agent (disease risk) value, one useful tool is the package "igraph". It allows for a self-designed network structure. More details about "igraph" can be found in the tutorial (https://igraph.org/r/). Run the demo code to visualize the recovered baseline network (e.g., the non-smoking group), where the agent is changing from no risk to high risk.
   b. Covariate-effect networks—Run the demo code for visualizing three recovered covariate-effect (e.g., effect caused by smoking) networks, where the agent is changing from no risk to low/moderate/high risk (data-driven or subjective cut-points selected for risk levels).
3. Visualize the interaction behaviors for a given gene over the spectrum of the agent
   Select one specific gene of interest, and run the demo code to plot the trend effect and interaction effects over the whole spectrum of the agent.

Data analysis

The goal of this section is to apply the procedure detailed above to learn and visualize the hypertension risk-associated pseudo-dynamic gene–gene networks in blood vessel tissue. The example data for illustrating the above procedure is from the GTEx project and consists of gene expression data from blood vessels and de-identified subsamples of phenotypic information from donors. Note that the raw phenotype data is confidential and protected. The illustrating data used in this protocol is de-identified and a subset of the raw data. The detailed information on data analysis is in the original paper (Chen et al., 2022), as well as in our GitHub repository. A summary is listed below.

1. Use the phenotype data (named "pheno_used") to predict the risk of having hypertension as the agent. The predicted disease risk can be found in Tutorial 1 in the GitHub repository.
2. Based on the imputed disease risk of having hypertension, use the gene expression data (named "data vessel") to fit the varying coefficient model, construct the base matrices, and then train the network learning model.
3. Visualize the baseline network (the non-smoking group), as the risk of having hypertension is changing from no risk to high risk (figure 1 in Tutorial 3 in the GitHub repository and figure 3 in the original paper). Visualize the smoking-effect-associated networks, as the risk of having hypertension is changing from no risk to low risk, moderate risk, and high risk (figure 2 in Tutorial 3 in the GitHub repository and figure 3 in the original paper).
4. Additionally, we pick the C3orf70 gene to visualize interaction behaviors over the whole spectrum of the risk of having hypertension for illustration. This gene is selected in a previous screening step and was shown to be associated with hypertension and neural and neurobehavioral development in literature (Samblas et al., 2019). In figures 3–6 in the GitHub repository, smoking has a negative effect on the expression of the target gene, and CTSD, GALNT4, NDUFA4L2, and RCN3 genes are detected as inhibiting the expression of the target gene.

Acknowledgments

Wu's work was partially supported by Grant U01 HL119178 from the National Heart, Lung and Blood Institute (NHLBI) and 5R01HD086911-02 from the National Institute of Child Health and Human Development (NICHD), the National Institute of Health. Wang's research was partially supported by Grants KL2 TR000126 and TR002015 from the National Center for Advancing Translational Sciences (NCATS) and start-up funding from Case Western Reserve University. Yu's work was supported by the Fundamental Research Funds for the Central Universities of China (Grant No. JZ2022HGQA0151). The content is solely the responsibility of the authors.
The original research paper from which this protocol was derived: Chen et al. (2022) published in Bioinformatics.

Competing interests

The authors declare that no competing interests exist.

References

Chen, C., Jiang, L., Fu, G., Wang, M., Wang, Y., Shen, B., Liu, Z., Wang, Z., Hou, W., Berceli, S. A. and Wu, R. (2019). An omnidirectional visualization model of personalized gene regulatory networks. *NPJ Syst Biol Appl* 5: 38.

Chen, C., Jiang, L., Shen, B., Wang, M., Griffin, C. H., Chinchilli, V. M. and Wu, R. (2021). A computational atlas of tissue-specific regulatory networks. *Front Syst Biol* https://doi.org/10.3389/fsysb.2021.764161.

Chen, C., Shen, B., Ma, T., Wang, M. and Wu, R. (2022). A statistical framework for recovering pseudo-dynamic networks from static data. *Bioinformatics* 38(9): 2481-2487.

Griffin, C., Jiang, L. and Wu, R. (2020). Analysis of quasi-dynamic ordinary differential equations and the quasi-dynamic replicator. *Physica A* 555: 124422.

Consortium, G. T. (2020). The GTEx Consortium atlas of genetic regulatory effects across human tissues. *Science* 369(6509): 1318-1330.

Malatras, A., Michalopoulos, I., Duguez, S., Butler-Browne, G., Spuler, S. and Duddy, W. J. (2020). MyoMiner: explore gene co-expression in normal and pathological muscle. *BMC Med Genomics* 13(1): 67.

Oliva, M., Munoz-Aguirre, M., Kim-Hellmuth, S., Wucher, V., Gewirtz, A. D. H., Cotter, D. J., Parsana, P., Kasela, S., Balliu, B., Vinuela, A., et al. (2020). The impact of sex on gene expression across human tissues. *Science* 369(6509): eaba3066.

Saha, A., Kim, Y., Gewirtz, A. D. H., Jo, B., Gao, C., McDowell, I. C., Consortium, G. T., Engelhardt, B. E. and Battle, A. (2017). Co-expression networks reveal the tissue-specific regulation of transcription and splicing. *Genome Res* 27(11): 1843-1858.

Wu, R. and Jiang, L. (2021). Recovering dynamic networks in big static datasets. *Phys Rep* Volume 912: 1-57.

Samblas, M., Milagro, F. I. and Martinez, A. (2019). DNA methylation markers in obesity, metabolic syndrome, and weight loss. *Epigenetics* 14(5): 421-444.