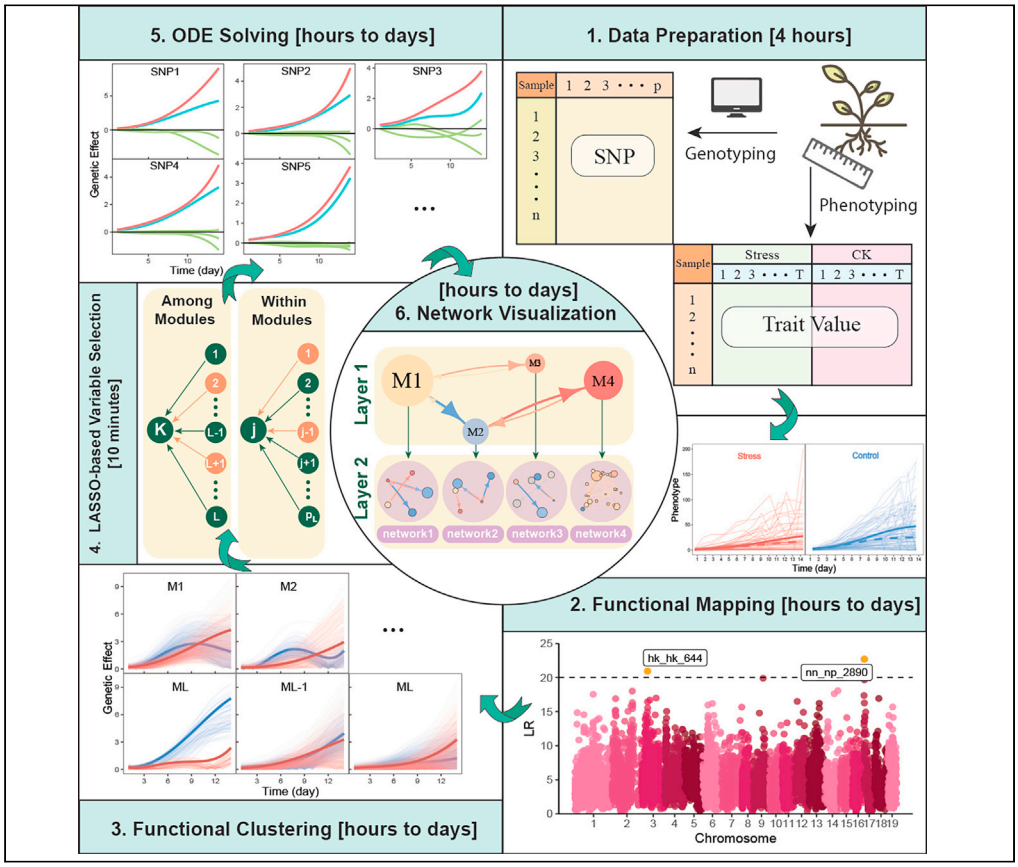
STAR Protocols

CellPress
OPEN ACCESS

Protocol

FunGraph: A statistical protocol to reconstruct omnigenic multilayer interactome networks for complex traits



We describe a statistical protocol of how to reconstruct and dissect functional omnigenic multilayer interactome networks that mediate complex dynamic traits in a genome-wide association study (GWAS). This protocol, named FunGraph, can analyze how each locus affects phenotypic variation through its own direct effect and a complete set of indirect effects due to regulation by other loci co-existing in large-scale networks. FunGraph is applicable to any GWAS aimed to characterize the genetic architecture of dynamic phenotypic traits.

Ang Dong, Li Feng, Dengcheng Yang, Shuang Wu, Jinshuai Zhao, Jing Wang, Rongling Wu

rwu@phs.psu.edu

Highlights

TurboID enabled biotin-based proximity labeling protocol for *C. elegans*

Experimental design guidelines for proximity labeling in *C. elegans*

A step-by-step TurboID protocol from transgene design to protein identification

Dong et al., STAR Protocols *2*, 100985
December 17, 2021
https://doi.org/10.1016/j.xpro.2021.100985

STAR Protocols

CellPress
OPEN ACCESS

Protocol

FunGraph: A statistical protocol to reconstruct omnigenic multilayer interactome networks for complex traits

Ang Dong,[1,3] Li Feng,[1] Dengcheng Yang,[1] Shuang Wu,[1] Jinshuai Zhao,[1] Jing Wang,[1] and Rongling Wu[1,2,4,*]

[1]Center for Computational Biology, College of Biological Sciences and Technology, Beijing Forestry University, Beijing 100083, China

[2]Center for Statistical Genetics, Departments of Public Health Sciences and Statistics, The Pennsylvania State University, Hershey, PA 17033, USA

[3]Technical contact

[4]Lead contact

*Correspondence: rwu@phs.psu.edu
https://doi.org/10.1016/j.xpro.2021.100985

SUMMARY

We describe a statistical protocol of how to reconstruct and dissect functional omnigenic multilayer interactome networks that mediate complex dynamic traits in a genome-wide association study (GWAS). This protocol, named FunGraph, can analyze how each locus affects phenotypic variation through its own direct effect and a complete set of indirect effects due to regulation by other loci co-existing in large-scale networks. FunGraph is applicable to any GWAS aimed to characterize the genetic architecture of dynamic phenotypic traits.
For complete details on the use and execution of this protocol, please refer to Wang et al. (2021).

BEFORE YOU BEGIN

Complex traits are of paramount importance to many fields of modern agriculture, biology, and biomedicine, but are also very difficult to study because of their complex genetic architecture. Traditional approaches based on reductionist thinking can identify individual key quantitative trait loci (QTLs) and have been instrumental for the identification of major QTLs for a variety of complex traits (Bradbury et al., 2007; Burga et al., 2019; Thavamanikumar et al., 2013). Because each complex trait needs a time to express, a mapping approach that captures the dynamic feature of complex traits, named as functional mapping (FunMap), has been developed and applied in a variety of genetic studies (Ma et al., 2002; Wu and Lin, 2006; Wang et al., 2019; Li and Sillanpää, 2015; Camargo et al., 2018). Thanks to the integration of biological principles underlying trait formation through mathematical equations, FunMap has proven itself to be biologically more relevant and statistically more powerful for QTL detection (Camargo et al., 2018; Liu et al., 2010; Lyra et al., 2020). However, increasing evidence shows that complex traits are controlled by a complete set of genes carried by an organism (Boyle et al., 2017). Thus, the best way to map complex traits is to coalesce all genes into an informative network that code all possible gene-gene interactions (Sun et al., 2021; Wu and Jiang, 2021). As an extension of FunMap, Wang et al. (2021) have more recently proposed a statistical method for reconstructing omnigenic multilayer interactome networks for dynamic traits from any large number of SNPs in a genetic mapping or association study.

Here, we describe a detailed protocol for Wang et al.'s method, making it more accessible to general geneticists. We name this protocol FunGraph as the extension of FunMap to draw a more

complete picture of genetic control mechanisms underlying complex traits. FunGraph includes a series of computational steps towards genetic mapping of complex traits, i.e., dynamic fitting of traits measured across time and space, FunMap of dynamic traits to detect significant QTLs and estimate genetic effect curves for each SNP, functional clustering of all SNPs into distinct modules based on their spatiotemporal similarity of genetic effect patterns, variable selection implemented to choose a set of the most significant SNPs that link with a given SNP, building and solving a system of nonlinear prey-predator ordinary differential equations (nLV ODEs), and reconstructing genetic networks using ODE parameters. We show each step by illustrating the results from a GWAS experiment of Euphrates poplar. Multilayer interactome networks inferred by FunGraph provide a tool to characterize the genetic architecture of dynamic complex traits.

Download and install required software and R packages

Timing: [1 min]

FunGraph package is available from github (see key resources table). Basic knowledge about R scripting and modeling is required to understand this protocol. The following example (including data and scripts) is used to demonstrate the general framework of FunGraph.

To install FunGraph, first install R package devtools through command:

```
>install.packages("devtools")
```

then use the command:

```
>devtools::install_github("cxzdsa2332/FunGraph/FunGraph_0.1.0")
```

Troubleshooting 1

Alternatively, you can download the FunGraph_0.1.0.tar.gz file in the github repository and manually install FunGraph.

Before the FunGraph is used in R, the package importation is necessary by the following command:

```
>library(FunGraph)
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Genotype data for GWAS population	This protocol	N/A
Phenotypic data for GWAS population	This protocol	N/A
Software and algorithms		
R version 4.1.1	R Project (R Core Team 2020)	https://www.r-project.org/
RStudio version 1.4.1717	RStudio Team (2020)	http://www.rstudio.com/
FunGraph	This protocol	https://github.com/cxzdsa2332/FunGraph

(Continued on next page)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
mvtnorm R package version 1.1-2	Alan Genz, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, Torsten Hothorn (2021)	http://CRAN.R-project.org/package=mvtnorm
Orthopolynom R package version 1.0-5	Frederick Novomestky (2013)	https://CRAN.R-project.org/package=orthopolynom
ggplot2 R package version 3.3.5	Hadley Wickham (2016)	https://ggplot2.tidyverse.org
devtools R package version 2.4.2	Hadley Wickham, Jim Hester and Winston Chang (2021)	https://CRAN.R-project.org/package=devtools
igraph R package version 1.2.6	Csardi G, Nepusz T(2006)	https://igraph.org
glmnet R package version 4.1-2	Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010)	https://www.jstatsoft.org/v33/i01/
Other	a x86_64-w64-mingw32 platform with 16 Gb of memory, Intel Core i7-10700 processor and R version 4.1.1 as well as x86_64-pc-linux-gnu platform with 1Tb of memory, Intel Xeon CPU E7-8855 v4 processor and R version 3.6.3.	N/A

MATERIALS AND EQUIPMENT

FunGraph in this protocol were run and tested on a x86_64-w64-mingw32 platform with 16 Gb of memory, Intel Core i7-10700 processor and R version 4.1.1 as well as x86_64-pc-linux-gnu platform with 1Tb of memory, Intel Xeon CPU E7-8855 v4 processor and R version 3.6.3.

Pause Point: Each major calculation functions in FunGraph by default would write results to working directory for further analysis. However, it would be tedious to load every written result, user can simply save all temporary results at once:

```
>save.image("FunGraph.Rdata")
```

The previous session can be reloaded by command:

```
>load("FunGraph.Rdata ")
```

STEP-BY-STEP METHOD DETAILS

Data preparation

Timing: [4 h]

Before running FunGraph, user need to provide genotypic and phenotypic datasets, and they should be cleaned and merged to exactly the same format of the example data. Phenotypic dataset contains control group and the treatment group, each with same sample id as row names and same column number to represent the times phenotypic data were measured. Genotypic data have sample id as columns and SNP id as row names, and two additional columns "Linkage" and "Genetic_Distances(cM)" represent which linkage groups a certain SNP belongs and the position of SNPs in linkage groups (in centimorgan) respectively, while lacking of them did not affect the following calculation.

1. Organize the phenotypic data.

a. Remove or replace missing values in the data.
b. Log-transformed the phenotypic data by command log() if the number in the data vary widely is recommended, such as numbers differ by more than five order of magnitude.

2. Organize the genotype data.
   a. FunGraph only accept numeric value for calculation, SNP genotype data should be converted into 0,1,2 matrix.
   b. Replace missing value as 9.
3. Match the phenotypic and genotype data according to the same sample name. User can check the example genotypic and phenotypic datasets by following command:

```
>View(geno)

>View(pheno)
```

Functional mapping

Timing: [hours to days; computing resources and data size affect the overall timing]

Bivariate Functional mapping (biFunMap) is crucial to this model, for it excavates how specific QTLs determines the complex trait expressed in various environments. The mean vector and covariance structure should be modeled according to the design of the experiment, and general steps are described as follows:

4. First plot mean curve to check initial parameters by type command: Troubleshooting 2

```
>get_mean_curve_plot(

  pheno_df = pheno,

  times = 1:14,

  init_sd_par = c(0.95, 12, 1.02, 8))
```

pheno_df is phenotypic data, times is a vector of time point and init_sd_par is the initial parameters for biSAD covariance matrix

5. FunGraph already wrapped the mean curve modelling, covariance matrix modelling and likelihood ratio calculation into a function, use get_biFunMap_result to store these calculation results to an object named as "result1" or anything user prefer: Troubleshooting 3

```
>result1 <- get_biFunMap_result(

  geno_df = geno[,-1:-2],

  pheno_df = pheno,

  times = 1:14)
```

the input are the phenotypic and genotypic datasets, vector of time points and initial parameters for biSAD covariance matrix

6. The key of FunMap is modelling mean curve vector and covariance matrix by following two commands: Troubleshooting 4

a. Model the mean vector of the growth curve.

```
>get_mu(mu_par, times)
```

mu_par is a vector with five number corresponding to the unknown parameters in modified logistic growth curve, and times is a vector of time point

b. Model the covariance structure using command:

```
>get_biSAD1(par, n)
```

where par is the parameters to input, in this case par equal to init_sd_par we checked in previous step, and n is column number of the matrix.

7. After calculation is finished, visualization is carried out by following two functions:
   a. The manhattan plot.

```
>get_manh_plot(geno_df = geno, LR = result1$LR)
```

the input is genotypic data with additional information and LR values which was calculated and stored in "result1"

b. The genetic effect curve plot.

```
>get_genetic_effect_plot(
  genetic_effect = result1$genetic_effect,
  number = 10)
```

the input is calculated genetic effect data and the number of sub-plots for demonstration

Optional: Permutation tests can be used to determine the genome-wide critical threshold by command get_permutation, but it is extremely time-consuming.

Note: the mean vector of example script is modeled by modified logistic growth equation, and covariance structure by biSAD(1).

⚠ CRITICAL: Always uses get_mean_curve_plot to check the initial parameters are sound for optimization before running the whole FunMap step.

Functional clustering

Timing: [hours to days; computing resources and data size affect the overall timing]

According to modularity theory (Melo et al., 2016), bivariate functional clustering (biFunClu) was introduced to cluster genetic effects into different functional modules (Wang et al., 2012). A hybrid of the EM and simplex algorithms were implanted to obtain the functional modules.

By default, after running previous step, a file named "genetic_effect.csv" would be generated and shall be used in this step. Calculated genetic effect dataset is already stored in FunGraph, users can view this dataset by command: genetic_effect.

8. Perform bifunctional clustering, i.e., clustering SNPs based on their temporal genetic effects in two different environments
   a. Check the dataset for calculation, user can use dataset from previous step or manually provide it.

```
>genetic_effect <- result1$genetic_effect

>View(genetic_effect)
```

   b. Prepare all initial parameters for functional clustering.

```
>input <- get_init_par(

  data = genetic_effect,

  k = 5,

  legendre_order = 4,

  times = 1:14,

  init_SAD_par = c(1.06,0.25, 1.15, 0.18))
```

      data is the dataset for cluster computation, and k is the number of clusters wanted, legendre_order is the order of Legendre Polynomials, times is time points and init_SAD_par is initial parameters for biSAD covariance matrix
   c. Perform functional clustering by command: Troubleshooting 3

```
>c1<-get_cluster(input = input, itermax = 100)
```

      input is result from 8.a, and itermax control the maximum number of iteration in EM algorithm
   d. Take a look at classification result:

```
>get_cluster_base_plot(

  clustered_data = c1$clustered_data[[1]])
```

      the input clustered_data directly from the get_cluster result

9. Since we do not know the optimal number of clusters (k), we need to perform the above command for different values of k. For instance, by varying k from 1 to 10 clusters. The criteria of optimal k number was determined by BIC value, user can check it by command: Troubleshooting 5

```
>print(c1$BIC)
```

Note: The mean vector of example script is modeled by Legendre Polynomials of order 4, and covariance structure by biSAD(1).

⚠ CRITICAL: When the Log-likelihood value changed drastically (e.g. from −20000 to −1000) in Functional Clustering using function get_cluster, an early stop for iteration should be

made to prevent incorrect parameters estimation by setting "Delta" parameter in get_cluster function.

LASSO-based variable selection

Timing: [10 min]

By viewing all genes that function as a dynamic system, any one gene may interact with other genes. However, it is impossible that each gene interacts with every other gene to form a fully interconnected network because this does not assure the system to be robust in response to environmental change. FunGraph implements a LASSO-based procedure to choose a small set of the most significant genes that links with a given gene across time points.

10. Lasso-based variable selection to select the most significant relevant Modules/SNPs from Modules/SNPs for Module i/SNPs i.
    a. First calculate the relationship between modules.

```
>get_interaction(data, col, reduction = FALSE)
```

the input data is the genetic effect data of modules, col is the number of rows, and reduction = FALSE means usually dimensionality reduction is not needed in performing variable selection between modules.

    b. Then the variable selection is performed within modules.

```
>get_interaction(data, col, reduction = TRUE)
```

the input data is the genetic effect data of SNPs in a certain module

Optional: The reduction = TRUE option can be FALSE in 10.b if there are no need for dimensionality reduction (e.g. the number of SNPs with in module is almost as same as the number of modules, usually less than 100).

ODE solving

Timing: [hours to days; computing resources and data size affect the overall timing]

A system of nLV ODEs are formulated according to evolutionary game theory (Wang et al., 2021), with the independent part describing the inner genetic effect of SNP i and the dependent part describing the influential genetic effect of other SNPs. Thereafter, the genetic network could be reconstructed through the decomposition of net genetic effect of each SNP.

11. Genetic network reconstructed for modules.
    a. Prepare genetic effect dataset for modules

```
>module_data <- get_module_data(
  data_par = c1$curve_par,
  times = 1:14)
```

where data_par is the parameters of Legendre Polynomials to model mean genetic effect curve, and times is the time points

b. Solve ODE between modules. Troubleshooting 6

```
>module_ode1 <- get_ode_par(
  data = module_data[[1]],
  times = 1:14,
  order = 3,
  reduction = FALSE,
  parallel = FALSE)
```

the input data is genetic effect matrix, times is time points, order is Legendre Polynomials order used in genetic effect decomposition, reduction is whether to use dimensionality reduction, and parallel is whether to use parallel calculation

c. The result from 11.b need to be further processed.

```
>module_net1 <- get_all_net(module_ode1)
```

the input is the result from 11.b

d. Now the ODE result can be plotted as decomposition of genetic effect curve: Troubleshooting 7

```
>get_decomposition_plot(
  input1 = module_ode1, input2 = module_net1,
  i = 1)
```

this function needs 11.b and 11.c result, the third parameters i indicates which module used in plot

e. Calculate maximum effect to control color used in network plot:

```
>max_effect <- cbind(
  get_max_effect(module_net1),
  get_max_effect(module_net1))
```

get_max_effect use the result of get_all_net to calculate the maximum genetic effect value

f. Finally, network can be reconstructed by command:

```
>network_plot(
  k = module_net1,
  title = 'CK',
  max_effect = max_effect,
  save_plot = FALSE)
```

k is the result of get_all_net, title indicate what plot title user want, max_effect directly from previous step and save_plot control whether to save PDF file.

12. Genetic network reconstructed for SNPs is very similar to previous step.
    a. Extract genetic effect data of a module.

```
>m1_ck <- get_subset_data(
  data = c1$clustered_data[[1]],
  cluster = 1 )
```

    get_subset_data function selects the subset cluster by input the functional clustering result.
    b. Follow every step in 11.b-f, remember to use "<-" to assign result for SNPs to a different new name.

Optional: get_net_output can be used to export network attributes for Cytoscape visualization.

Reconstructing multilayer interactome networks

Timing: [hours to days; computing resources and data size affect the overall timing]

The salient feature of FunGraph is to organize hundreds of thousands or thousands of thousands of SNPs in a GWAS into a multilayer interaction network by classifying these SNPs into distinct modules. At the first layer is the interaction network among modules, reconstructed from the mean genetic effect curve of all SNPs within modules. The second-layer network is reconstructed from genetic effects curves of individual SNPs from a module. SNP networks, nested within a module, can map the fine-grained (microscopic) architecture of genetic interactions. In practice, if the size of a module is still too large for reconstructing its SNP network, we can further classify it into its submodules. Similarly, we classify a submodule into its multiple sub-submodules, and this procedure is repeated until the size of a unit is maneuverable.

13. Generally, a module that contains more than 500 SNPs is difficult for network visualization and should be further classified, user can check number of SNPs with in module by command:

```
>table(table(c1$clustered_data[[1]]$cluster) )
```

14. For a target module, classification and ODE solving can be easily done through abovementioned approaches.
    a. Select a module for further FunMap process.

```
>m1_ck <- get_subset_data(
  data = c1$clustered_data[[1]],
  cluster = 1 )
>input2 <- get_init_par(
  data = m1_ck,
  k = 3,
  legendre_order = 4,
  times = 1:14)
>c2 <- get_cluster(input = input2)
```

b. Solve the ODE and reconstruct network for submodules.

```
>submodule1_data <- get_module_data(
  data_par = c2$curve_par,
  times = 1:14)
>submodule1_ode1 <- get_ode_par(
  data = submodule1_data [[1]],
  times = 1:14,
  order = 3)
>submodule1_net1 <- get_all_net(submodule1_ode1)
>max_effect1 <- cbind(
  get_max_effect(submodule1_net1),
  get_max_effect(submodule1_net1))
>network_plot(
  k = submodule1_net1,
  title = ''Submodule1_CK'',
  max_effect = max_effect1)
```

c. Similarly, the SNP network can be reconstruction by the same functions.

EXPECTED OUTCOMES

The major calculation function get_biFunMap_result in Functional Mapping part should generate an R list object, involving overall curve fitting, LR scores, a set of estimated logistic growth equation parameters, and genetic effect curves for each SNP. Together with the plot functions get_mean_curve_plot, get_manh_plot and get_genetic_effect_plot, the results can be further plotted as Figure 1.

The function get_cluster of Functional Clustering part should generate an R list object, involving all estimated parameters, the BIC value, and the classified modules of SNPs, by a slight modification of get_cluster_base_plot the results of Functional Clustering are showed in Figure 2.

The function in ODE solving part already includes LASSO-based variable selection. An example variable selection result between modules is given by function get_interaction in vignette of FunGraph. The result R list contains the name of module, the coefficients of LASSO regression, and the relevant modules (Figure 3).

The get_ode_par funciton in ODE solving step would generate a list contain parameters of Legendre polynomials, variable selection results and some useful information. These outcomes can be plugged into get_decomposition_plot and network_plot functions for further visualization (Figure 4).

From the result of FunMap, significant loci "nn_np_2890" was chosen to demonstrate multilayer interactome networks. This SNP belongs to module 13 which contains 548 SNPs, thereby module 13 was further classified into 8 submodules and a three-layer networks were constructed (Figure 5).

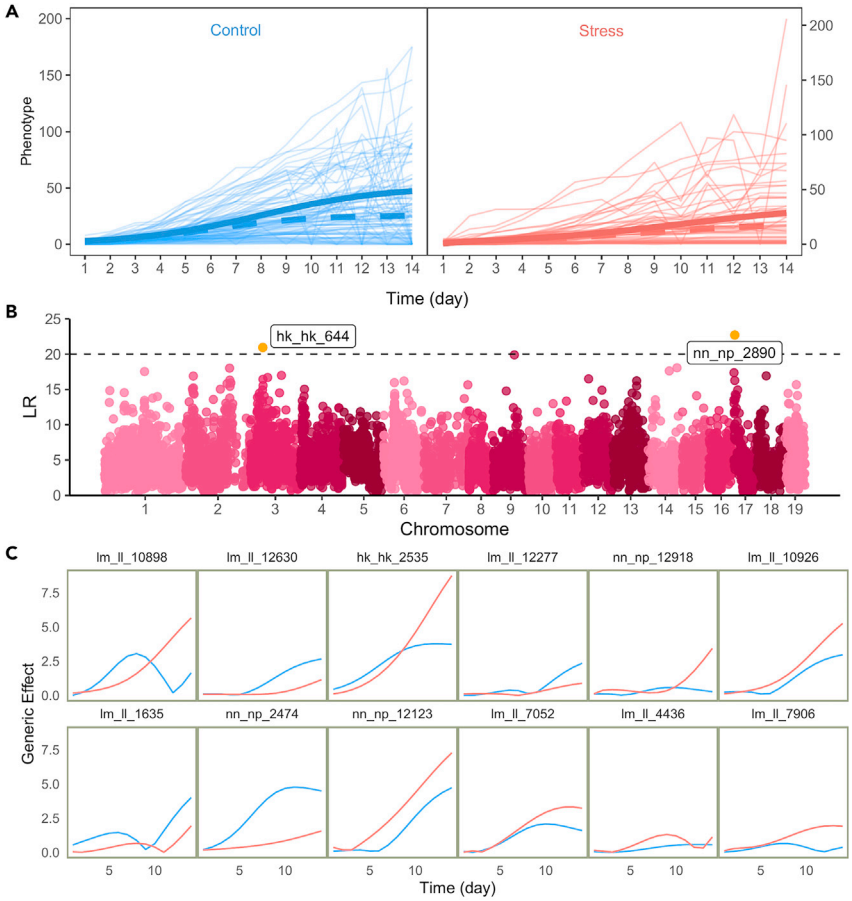


Figure 1. The result of FunMap

(A) get_mean_curve_plot function shows phenotypic data of roots fitted by a modified logistic growth equation cultured in salt-free (control) and salt-exposed (stress) media. Thick line are the mean growth trajectories of all individuals.

(B) get_manh_plot plot the significance tests for SNPs across the whole chromosome by biFunMap. SNPs above the dashed line are considered as significant loci that affect root growth.

(C) get_genetic_effect_plot generates randomly selected genetic effect curves of 12 SNPs under control (blue) and stress condition (red).

LIMITATIONS

Multilayer interactome networks by FunGraph are reconstructed on the basis of dynamic genetic effects estimated from longitudinal data by FunMap. In practice, many genetic mapping or GWAS experiments do not measure phenotypic traits repeatedly over a series of time points. Thus, it is impossible to reconstruct multilayer networks for these experiments unless a new statistical model is developed to accommodate the static features of these data.

The precision of network reconstruction depends on the number of time points (for ODE solving) and the estimation precision of dynamic genetic effects. If it is challenging to obtain high-density time points required for precise effect estimation, efforts should be made to increase the precision of trait phenotyping. For example, by producing multiple replicates, measurement noise can reduce, leading to increased phenotyping precision and heritability.

FunGraph is based on the absolute size of overall genetic effects. However, genetic effects at individual loci can be better described by additive and/or dominant effects at individual SNPs, which

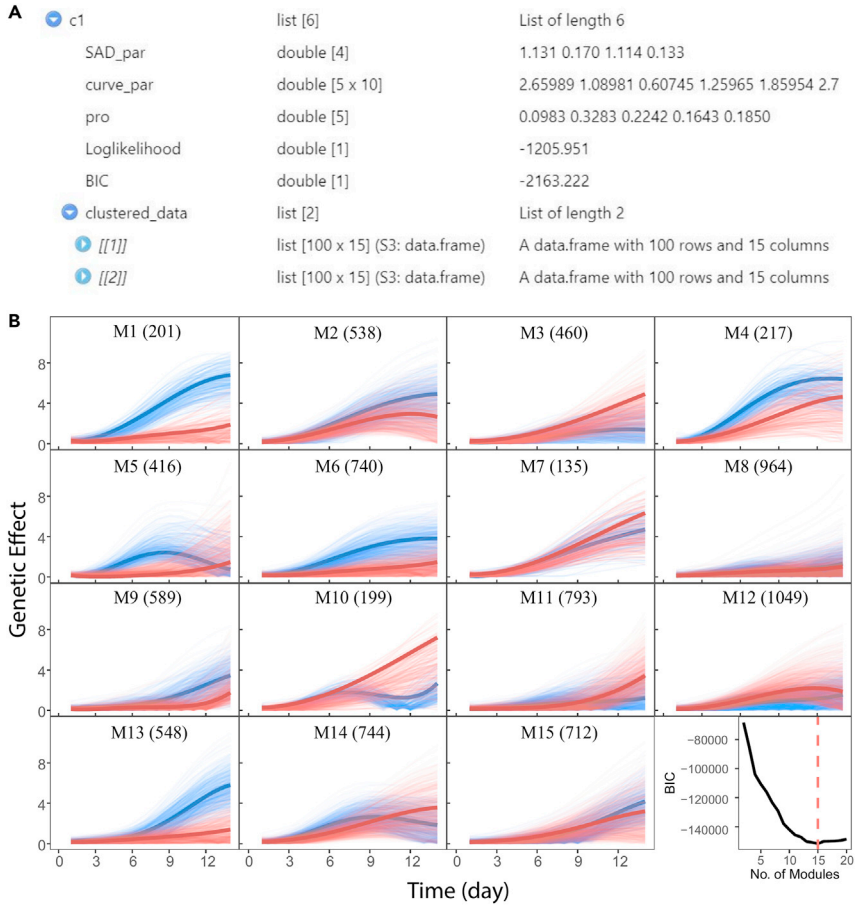


Figure 2. The result of functional clustering

(A) Screenshot of output list object from get_cluster function with $L = 5$.

(B) Classification results of genetic effect curves under control (blue) and stress conditions (red). BIC analysis detects 15 as the optimal number of modules (L)

may be positive or negative. How to incorporate both the magnitude and sign of genetic effects into nLV-based ODEs is not a trivial issue, but need to be resolved for better characterizing the genetic architecture of complex traits.

The current FunGraph package only contain limited functions to model mean curve, covariance matrix and control and the treatment groups are required, which did not cover the full application range of FunGraph. Besides, the numeric optimization may meet difficulty when dealing with large dataset, when cluster number k in Functional Clustering step is large (e.g., k = 100) unexpected error may occur. More features in FunGraph package and better initial parameters choice, parameters estimation may add in future.

TROUBLESHOOTING

Problem 1

When installing FunGraph, R return with error: dependencies 'xxx', 'xxx' are not available for package 'FunGraph'

Potential solution

Manually install missing dependencies packages 'xxx' through command:

```
>install.packages('xxx')
```

Problem 2
When running FunGraph, R return with error: singular gradient/initial value in 'vmmin' is not finite/non-finite value supplied by optim

Potential solution
Usually it was caused by redundant information in dataset, just remove non-numeric content in datasets.

Alternatively, column names of input dataset contain underscore, hash, dash and so on may results failure in matching column names in get_ode_par function.

Another possible solution is to try different value for initial parameters, generally initial parameters for biSAD covariance matrix should between 0.1 to 10, and initial parameters for model logistic growth curve should be 0.1 to maximum observation phenotypic data.

Problem 3
The calculation took too much time and/or estimated parameters are inaccurate (Table 1).

Potential solution
Parameters estimation for biFunMap and biFunClu can be improved through the manually given determinant and inverse for biSAD1 covariance matrix than the implanted solve() and det() functions in R, but it would be challenging and time consuming.

Problem 4
The value of BIC kept fluctuating with the increasement of k, therefore the optimal number of k is difficult to choose.

Potential solution
The initial parameters were randomly given and may influent the outcome of BIC value depending on the dataset, for a certain k it is recommended to run several times and choose the minimal BIC value as the actual BIC value.

Problem 5
Calculation stopped when running get_ode_par.

Potential solution
In the rare case that no connection can be found between target module/SNP and the rest dataset, get_interaction function would return with missing values NA. Users can either manually assign a most relevant dataset by cor() function, or skip this module/SNP

Problem 6
The intrinsic growth curves of certain genetic effect may not be positive (The estimated genetic effect data are all positive).

Potential solution
Try different ODE initial values in get_value funciton, regularization in estimating ODE parameters by modify ode_optimize function in source R code should also be a potential solution.

QUANTIFICATION AND STATISTICAL ANALYSIS

FunGraph is the extension of FunMap to reconstruct omnigenic interactome networks. As illustrated in the Graphic Abstract, FunGraph is constructed by several key steps as follows: (1) associating

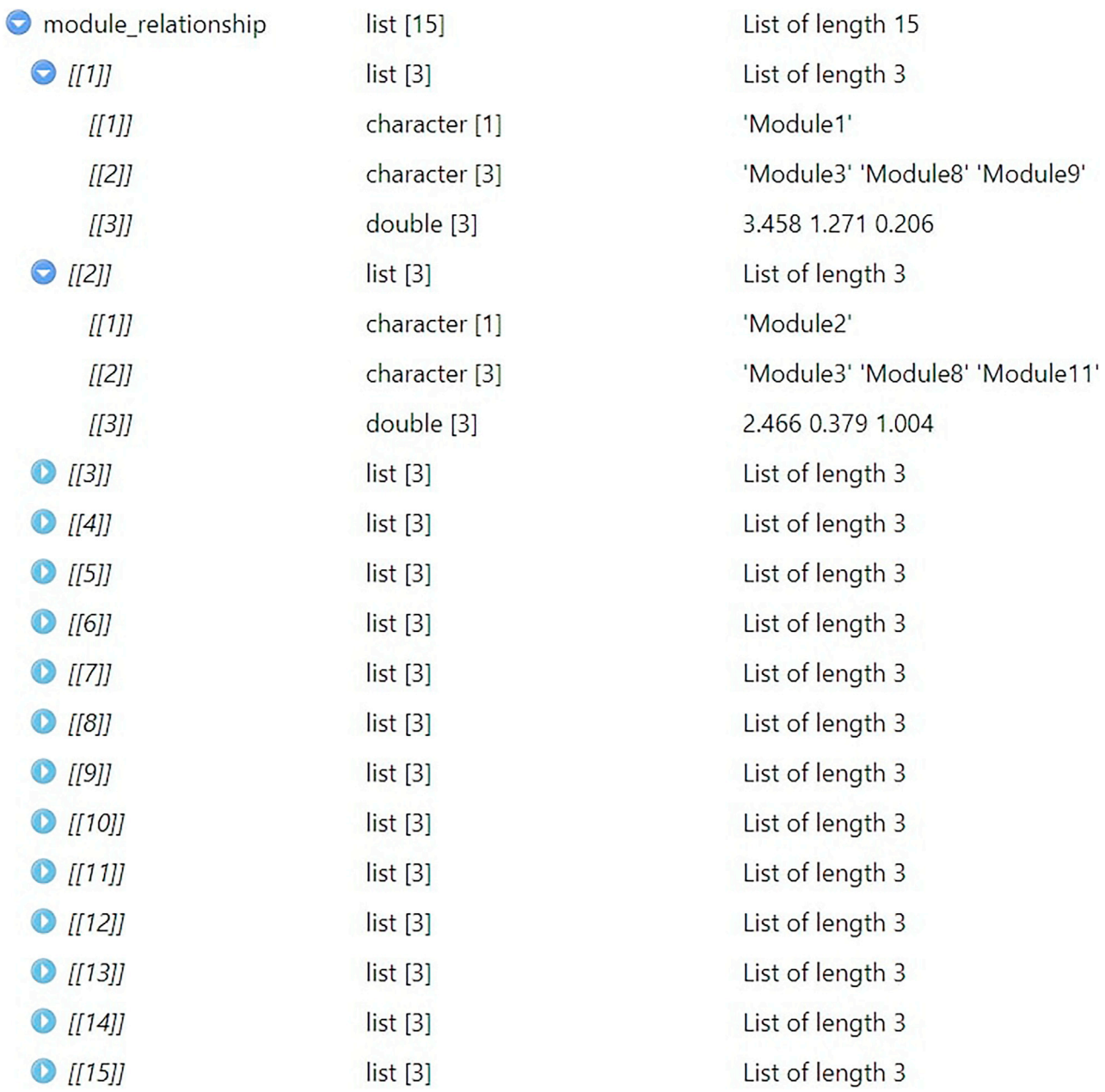


Figure 3. The result of LASSO-based variable selection

Screenshot of get_interaction result for LASSO-based variable selection (control condition data used).

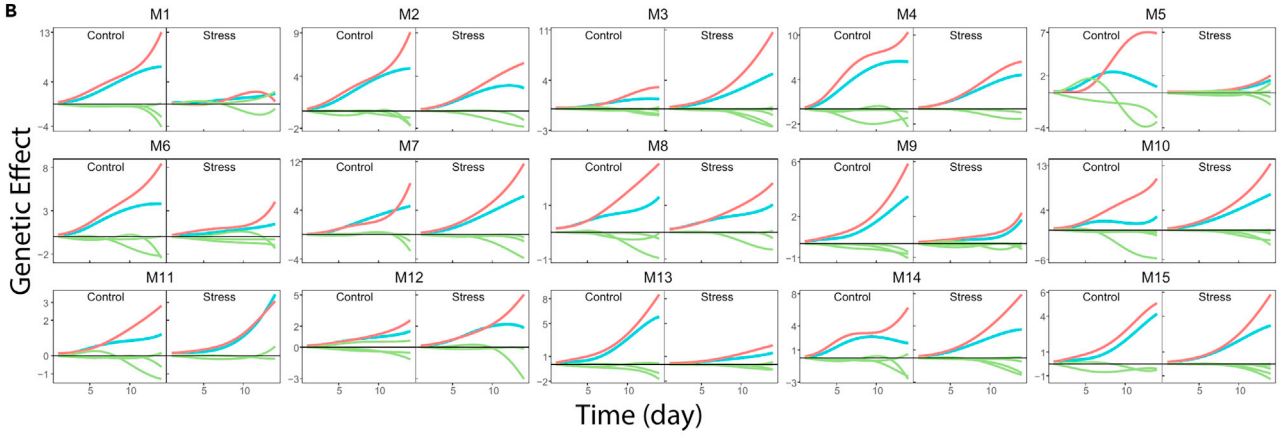
genotype data with phenotype data via FunMap, in which the temporal pattern of genetic effects exerted by each SNP is illustrated and significant SNPs (QTLs) are identified and annotated, (2) functional clustering of all SNP into distinct modules based on the similarity of their genetic effect patterns, where an optimal number of modules is determined according to information criteria, such as BIC, (3) LASSO-based variable selection that identifies a small set of the most significant entities (modules or SNPs) that link with a given entity, (4) solving ODEs that characterize independent and dependent genetic effects of each entity through which the estimated ODE parameters are used to describe the magnitudes and/or signs of these two effect components, and (5) reconstructing a multilayer omnigenic interactome network using graph software. The SNP-SNP interaction

A

Name	Type	Value
m1_ck_net	list [9]	List of length 9
[[1]]	list [6]	List of length 6
[[1]]	character [1]	'hk_hk_2636'
[[2]]	character [1]	'hk_hk_2796'
[[3]]	double [3]	0.144 -1.455 -0.157
[[4]]	double [3]	-0.701 1.397 0.397
[[5]]	double [2]	3.438 -0.731
[[6]]	double [14 x 2]	0.230 0.525 0.971 1.6
[[2]]	list [6]	List of length 6

Name	Type	Value
m1_ck_ode	list [5]	List of length 5
lop_par	list [9]	List of length 9
relationship	list [9]	List of length 9
dataset	list [9 x 14] (S3: data.frame)	A data.frame wit
times	integer [14]	1 2 3 4 5 6 ...
order	double [1]	3

B



C

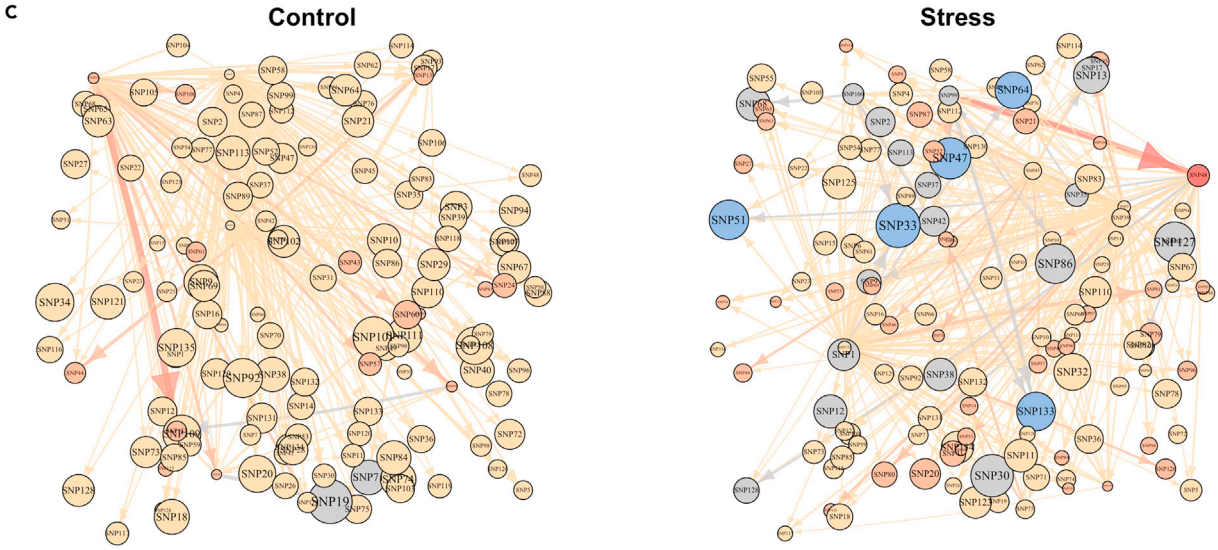


Figure 4. The result of ODE solving

(A) Screenshot of get_ode_par and get_all_net results.

(B) The combined plot returned by function get_decomposition_plot. Every net genetic effect of a certain module (SNPs) can be decomposed into its independent effect (red line) and dependent effects (green lines) received from other modules (SNPs).

(C) The microscopic genetic network reconstructed for 135 SNPs in module 7 via command network_plot. The sizes of the circles equal to the total regulatory effect received. Arrow lines denote the interaction between SNPs, with thickness proportional to the strength of the interaction. Red lines and blue lines denote the up-regulation and down-regulation of one SNP for the next, respectively.

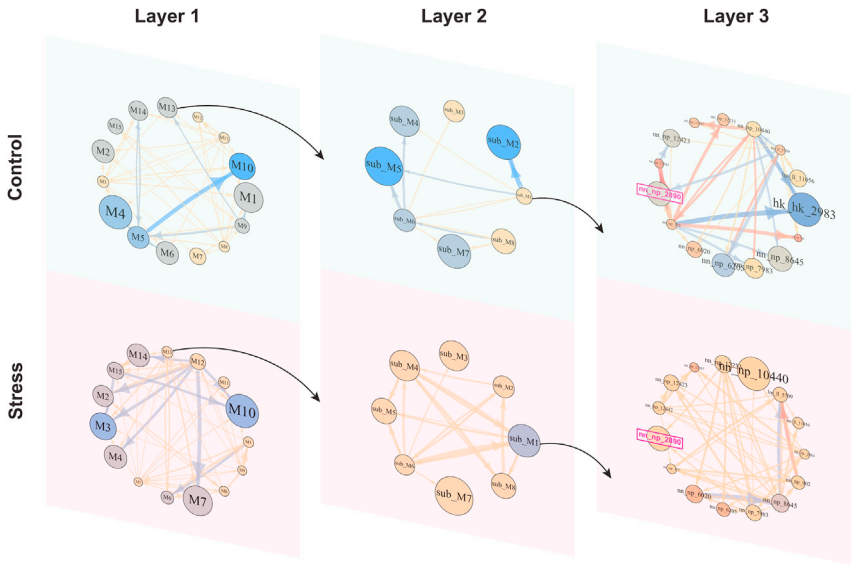


Figure 5. The result of multilayer interactome networks. The first layer is the interaction network among modules, the second layer submodule network reconstructed from genetic effects curves of individual SNPs from module 13, and the third layer shows the microscopic SNP interaction within submodule 1.

network codes a detailed roadmap of how each SNP (regardless of its significance according to FunMap) interacts with every other SNP to mediate phenotypic variation.

RESOURCE AVAILABILITY

Lead contact

Further information and request of resources should be directed to Rongling Wu (rwu@bjfu.edu.cn).

Materials availability

This study did not generate new unique reagents.

Data and code availability

All data, analysis, and modeling code have been deposited to github: https://github.com/cxzdsa2332/FunGraph

ACKNOWLEDGMENTS

We thank the colleagues at the Center for Computational Biology at Beijing Forestry University for their contributions to this work.

AUTHOR CONTRIBUTIONS

A.D. wrote code, analyzed the data, and wrote the manuscript. L.F., D.Y., S.W., J.Z., and J.W. participated in model derivations and data analysis. R.W. supervised the project and wrote the manuscript with A.D.

Table 1. The inaccurate inverse of a AR(1) covariance matrix (σ^2 = 2, ρ=0.4, t = 5) by default solve() function in R, number with * should be 0

1	−0.4	−2.33E−17*	0	3.19E−18*
−0.4	1.16	−0.4	0	2.78E−18*
9.63E−35*	−0.4	1.16	−0.4	2.78E−17*
−1.39E−18*	2.66E−17*	−0.4	1.16	−0.4
3.47E−18*	2.78E−18*	5.55E−17*	−0.4	1

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Boyle, E.A., Li, Y.I., and Pritchard, J.K. (2017). An expanded view of complex traits: from polygenic to omnigenic. Cell *169*, 1177–1186.

Bradbury, P.J., Zhang, Z., Kroon, D.E., Casstevens, T.M., and Buckler, E.S. (2007). TASSEL: software for association mapping of complex traits in diverse samples. Bioinformatics *23*, 2633–2635.

Burga, A., Ben-David, E., Vergara, T.L., Boocock, J., and Kruglyak, L. (2019). Fast genetic mapping of complex traits in *C. elegans* using millions of individuals in bulk. Nat. Commun. *10*, 1–10.

Camargo, A.V., Mackay, I., Mott, R., Han, J., Doonan, J.H., Askew, K., Corke, F., Williams, K., and Bentley, A.R. (2018). Functional mapping of quantitative trait loci (QTLs) associated with plant performance in a wheat MAGIC mapping population. Front. Plant Sci. *9*, 887.

Li, Z., and Sillanpää, M.J. (2015). Dynamic quantitative trait locus analysis of plant phenomic data. Trends Plant Sci. *20*, 822–833.

Liu, G., Li, M., Wen, J., Du, Y., and Zhang, Y.-M. (2010). Functional mapping of quantitative trait loci associated with rice tillering. Mol. Genet. Genom. *284*, 263–271.

Lyra, D.H., Virlet, N., Sadeghi-Tehran, P., Hassall, K.L., Wingen, L.U., Orford, S., Griffiths, S., Hawkesford, M.J., and Slavov, G.T. (2020). Functional QTL mapping and genomic prediction of canopy height in wheat measured using a robotic field phenotyping platform. J. Exp. Bot. *71*, 1885–1898.

Ma, C.-X., Casella, G., and Wu, R. (2002). Functional mapping of quantitative trait loci underlying the character process: a theoretical framework. Genetics *161*, 1751–1762.

Melo, D., Porto, A., Cheverud, J.M., and Marroig, G. (2016). Modularity: genes, development, and evolution. Annu. Rev. Ecol. Evol. Syst. *47*, 463–486.

Sun, L., Dong, A., Griffin, C., and Wu, R. (2021). Statistical mechanics of clock gene networks underlying circadian rhythms. Appl. Phys. Rev. *8*, 021313.

Thavamanikumar, S., Southerton, S.G., Bossinger, G., and Thumma, B.R. (2013). Dissection of complex traits in forest trees—opportunities for marker-assisted selection. Tree Genet. Genomes *9*, 627–639.

Wang, H., Ye, M., Fu, Y., Dong, A., Zhang, M., Feng, L., Zhu, X., Bo, W., Jiang, L., and Griffin, C.H. (2021). Modeling genome-wide by environment interactions through omnigenic interactome networks. Cell Rep. *35*, 109114.

Wang, H., Zhang, F., Zeng, J., Wu, Y., Kemper, K.E., Xue, A., Zhang, M., Powell, J.E., Goddard, M.E., and Wray, N.R. (2019). Genotype-by-environment interactions inferred from genetic effects on phenotypic variability in the UK Biobank. Sci. Adv. *5*, eaaw3538.

Wang, Y., Xu, M., Wang, Z., Tao, M., Zhu, J., Wang, L., Li, R., Berceli, S.A., and Wu, R. (2012). How to cluster gene expression dynamics in response to environmental signals. Brief. Bioinform. *13*, 162–174.

Wu, R., and Jiang, L. (2021). Recovering dynamic networks in big static datasets. Phys. Rep. *17*, 1–57.

Wu, R., and Lin, M. (2006). Functional mapping—how to map and study the genetic architecture of dynamic complex traits. Nat. Rev. Genet. *7*, 229–237.