Findr User Manual

Lingfei Wang
The Roslin Institute, University of Edinburgh

For version 0.5.0

Contents

1 Introduction

1.1 Basic information

Findr (Fast Inference of Networks from Directed Regulations) is a C library for the inference of gene regulatory networks. Interfaces for binary and python2, as well as a standalone package for R are provided for major functions. Findr performs causal inferences of pairwise gene regulations based on genotype and gene expression datasets or from gene expression dataset alone. Findr then bases on the causal inference results as prior edge information to construct a directed acyclic graph. A sparse Bayesian network can then be inferred with the R package lassopv to produce the p-value of every possible regulation [1]. Methodologies can be found in publication [2, 3]. The latest version of this documentation can be downloaded here.

1.2 Highlights

Findr provides accurate yet fast inferences of regulatory networks. In pairwise inference, Findr addresses two pitfalls of traditional causal inference (weak interactions and confounders) and is nearly 1,000,000 times faster than existing causal inference software (CIT [4]). In network reconstruction, Findr is also orders of magnitude faster than existing methods, such as bnlearn, with the resulting Bayesian network having comparable or better performances in terms of predictive power, inference accuracy, and FDR control.

For pairwise inference, Findr follows the inference strategy of Trigger [5], and performs multiple likelihood ratio tests with false positive rate estimation. The major difference is that Findr addressed the statistical and computational drawbacks of traditional causal inference, whose performance tends to get worse as the number of samples increase due to elevated false negative rate [2]. Findr has been demonstrated to overperform all other methods on the DREAM 5 Systems Genetics Challenge datasets. Findr also outperformed other inference methods in speed and accuracy in miRNA target prediction, include Pearson and Spearman correlations, mutual information, Lasso and elastic-net regressions, GENIE3 [6], CIT [4], etc. In addition, Findr performs meaningful false positive rate estimation, without being overly conservative as in traditional causal inference.

For network reconstruction, Findr uses prior edge information, such as Findr's own output from pairwise inference, to build a maximal directed acyclic graph with efficient algorithm [7]. The maximal directed acyclic graph can then be reduced to a sparse Bayesian network with proper variable selection [1, 3].

The acceleration of Findr originated from the statistical advances in [2] which have enabled analytical computation of null distributions, as well as multithreading with OpenMP and C language implementation with GSL (GNU Scientific Library).

1.3 License

Findr is licensed under GNU Affero General Public License, Version 3 (AGPL-3), which can be downloaded from https://www.gnu.org/licenses/agpl-3.0. GSL is separated licensed by its authors under GPL.

1.4 Contact

The authors welcome feedbacks in any form and perspective. No matter you have enquiries, suggestions, or critics, in installation, usage, future development, methodology, or collaboration, you can reach us at: Lingfei.Wang.github@outlook.com.

2 Findr library

Findr library is the core of computation for the software. The binary and python2 interfaces require Findr to be pre-installed, and call Findr during computation. The Findr R package contains a copy of Findr library by itself and does not need to install Findr library separately. Users of Findr R package can jump to Section 3.3.

2.1 Installation

The Findr library and its interfaces can function without installation after some adjustments. However we recommend installation following the steps below. If you encounter any errors during installation that can't be resolved independently, please contact us with detailed error messages.

- **Prerequisites:** Findr is written in C and by nature supports Windows, Linux, and Mac OS X platforms. The user also needs the following to build Findr library:
  - A recent GCC compiler that supports OpenMP interface.
  - An up-to-date GNU make utility.

  - GNU Scientific Library (GSL). GSL must be installed or placed in a searchable location for include and link.

- **Download:** Findr can be downloaded with git from github with the command:

```
git clone https://github.com/lingfeiwang/findr.git
```

  Alternatively it can also be downloaded from the webpage
  https://github.com/lingfeiwang/findr.

- **Customize makefile (optional):** You can customize how Findr is built and installed by editing its Makefile. This is useful for example when the user provides custom GSL location. Another possibility is the user does not have administrative priviledge on the computer, so a local installation is desired. In the latter case, the user must make sure the custom installation can be correctly located by binary and/or python2 interfaces, e.g. with environmental variables.

- **Build:** The source can be built with one line of command after downloading, in the directory it is downloaded or unzipped:

```
make
```

- **Install:** If you can gain superuser privilege, a normal install is recommended. This is also one line of command:

```
sudo make install
```

  Otherwise, the user is suggested to change install location by changing the 'PREFIX'. Then use the following command to install without superuser privilege:

```
make install
```

  The same PREFIX should then be set for binary interface. For details, see FAQ 3 in Section 6.

  We strongly advise against keeping multiple versions of Findr on the same machine, including installed and not installed versions, as this may confuse interfaces of which library to use.

- **Update:** To update Findr, simply download a newer version and install it to the same location. This will replace the older version.

- **Uninstall:** If you ever need to remove Findr from your machine permanently, this can be done with the command in the directory containing Findr source:

```
sudo make uninstall
```

  The 'PREFIX' variable should be adjusted beforehand if you want to uninstall from a custom location. Preferrably, the source code should have the same version as the installed library. Sudo may be needed depending on the location of installation.

2.2 General concepts

- **Library initialization:** The library must be initialized before called. During initialization, Findr initializes GSL, and processes the following initialization parameters:
  - loglv: The logging level of Findr. For details, see **Logging** below.
  - rs: Initial seed for random number generator. By default, uses the current time.
  - nth: Maximum number of parallel threads for computation. For best performance, it should not exceed the number of CPU cores. Its default value is the number of cores automatically detected, which is not always accurate.

- **Logging:** Findr has 13 logging levels:
  - Critical (0): The calculation has to halt due to critical error(s). Output is invalid.
  - Error (1-3): Part of function is lost in calculation due to uncritical error(s). Part of output is invalid.
  - Warning (4-6): Calculation is successful but Findr has encountered abnormal data. Findr attempted to correct anomalies but errors may have been introduced in output. User is advised to inspect their input.
  - Info (7-9): Running information of Findr.
  - Debug (10-12): More verbose running information.

  During initialization, the user is requested to input the logging level (loglv), so that only message levels not greater than loglv would printed. Findr library does not have a default log level, but in binary, python2 interfaces and R package, the default log level is 6.

2.3 Using Findr library

Findr library is well documented in its header files and source code. Developers are encouraged to build programs that call Findr, and to modify Findr freely for specific needs under current license.

3 Interfaces and packages

We provide binary and python2 interfaces for the major functions of Findr. Incompatible versions between interfaces and the library can create potential errors and limit reproducibility in data analysis. For this reason, users are advised to install/update interface(s) immediately after the library, and ensure they have the same version. Version differences are warned during interface execution. The R package of Findr includes the library of the same version, so the issue is absent.

The binary interface, python2 interface, and R package reveals the same set of functionality of Findr, under different naming and language conventions. User of a specific interface or package can proceed to installation and usage instructions of the respective section below, and then look for desired functions in Section 4.

3.1 Binary interface

The binary interface follows the same build, install, and uninstall steps as Findr library itself, after downloading with the command:

```
git clone https://github.com/lingfeiwang/findr-bin.git
```

or from the webpage https://github.com/lingfeiwang/findr-bin.

After installation, type 'findr' would execute the binary. However, if Findr library is installed in a custom location, user should change the Makefile of binary interface correspondingly, such as the 'PREFIX' variable. If the binary interface is installed in a custom location, the user needs to add the install location to 'PATH' environmental variable.

The binary interface only provides minimal validity checks on input data, although errors in calculation are notified in logs (see Section 2.2). Binary interface should be invoked as:

```
findr loglv rs nth method method_args...
```

The first three parameters are Findr library initialization parameters, as explained in Section 2.2 and Section 4.1. For automatic configuration, input zeros for all of them. method stands for the name of data analysis routine the user want to call, and method_args are its arguments which will be passed on to the method function. Available methods are listed in Section 4 in red.

Binary interface receives bulk data (vectors and row-major matrices) as input from files, and exports bulk output data to files. Both raw and tsv (tab separated values) input formats are accepted.

By default, raw format is used where float type data are in 32-bit raw format in native endianness, and genotype data are in 8-bit unsigned integer format. To use tsv format for all input and output files, append '_tsv' to the method name. Tsv format files should contain one row in each line, use tab (or space) as column separator, and not contain row or column names/indices.

3.2 Python2 interface

The python2 interface requires numpy, as it uses numpy.ndarray as input and output format. It also requires pre-installed Findr library. To install Findr's python2 interface, execute the following from command line (with proper privilege):

```
pip install findr
```

or download and install it from github with

```
git clone https://github.com/lingfeiwang/findr-python.git
cd findr-python
sudo python2 setup.py install
```

Alternatively, the python2 interface can be downloaded from https://github.com/lingfeiwang/findr-python.

To use Findr, first obtain an initialized library object with:

```
import findr
l=findr.lib(path=None,loglv=6,rs=0,nth=0)
```

path gives the exact location to search for Findr shared library in addition to default locations. Other three optional parameters are for library initialization and can be omitted for ordinary use. After above lines, Findr routines can be called as l's method. Exposed python functions of Findr are listed in Section 4, in which python method names are in green.

Bulk input and output data are formatted in numpy.ndarray for vectors and matrices, with 32-bit float for expression data and 8-bit unsigned integer for genotype data by default. Function returns are in dictionary type, in which each key contains an independent returned object. All functions share a returned key 'ret', which indicates a success in calculation with 0 and failure otherwise.

3.3 R package

Findr's R package is a standalone package which includes Findr library and GSL. It is locally compiled by *nix toolset, such as GCC and GNU make. On Mac OS and Linux, the R package can be downloaded and installed with

```
git clone https://github.com/lingfeiwang/findr-R.git
cd findr-R
R CMD INSTALL findr
```

Alternative, it can be downloaded at https://github.com/lingfeiwang/findr-R.

To use Findr, first initialize the library with:

```
library(findr)
findr.lib(loglv=6,rs=0,nth=0)
```

The three optional parameters are for library initialization and can be omitted for ordinary use. After above lines, Findr routines can then be called. Exposed R functions of Findr are listed in Section 4, in which R function names are in blue.

Bulk input and output data are formatted in R matrix or vector format, with double type for expression data and integer for genotype data. Function returns are in list type, in which each element contains an independent returned object.

4 Major functions

4.1 Library initialization

- Library initialization: Initializes the library and provides log level, initial random seed, and maximum thread count.
  (automatic through three initial parameters: loglv, rs, and nth, whose 0 indicates to use default values)
  l=findr.lib(path=None,loglv=6,rs=0,nth=0)
  findr.lib(loglv=6,rs=0,nth=0)

path	Extra location to search for Findr shared library in addition to default search paths. Only accepted in python2 interface.
loglv	Log level of library Findr. Only messages whose levels are not greater than the designated log level will be printed. Level 0: critical, 1-3: errors, 4-6: warnings, 7-9: information, 10-12: debug. Default value is level 6.
rs	Initial random seed. Default value means to use current time.
nth	Maximum number of parrallel threads in calculation. For best performance, this should not exceed the number of CPU cores. Default value indicates using the number of cores automatically detected, which is not always accurate.

4.2 Inference of pairwise regulations

To infer pairwise regulation ($A \rightarrow B$) based on pairwise information, gene expression levels of A and B and optionally genotype information of the best eQTL of A as $E(A)$ can be provided. With only expression data, pij_rank can be used. The inclusion of genotype information allows more complicated analysis, such as our novel method pij_gassist, or the traditional method pij_gassist_trad. Users have the freedom to customize the combination of tests by obtaining all five test results with pijs_gassist.

For best accuracy, it is advised to include as many secondary targets of the same type (e.g. gene expression levels) as possible, and then pick the ones of interest from Findr's output, instead of preselecting a minimal set of secondary targets. In this way, the accuracy of conversion from log likelihood ratios to probabilities are higher.

The exposed functions for pairwise regulation inference are listed below:

- Inference of correlation $A \cdots B$ probability with expression data only, by converting log likelihood ratios to probabilities per A. Methodology can be found in [2].
  pij_rank ft ft2 nt nt2 ns fp nodiag memlimit
  ans=l.pij_rank(dt,dt2,nodiag=False,memlimit=0)
  ans=findr.pij_rank(dt,dt2,nodiag=FALSE)

ft dt dt	Input matrix of expression levels of A, in file path, numpy.ndarray, or matrix format. Element [i,j] is the expression level of gene i of sample j. The matrix has dimension (nt,ns).
ft2 dt2 dt2	Input matrix of expression levels of B, in file path, numpy.ndarray, or matrix format. Element [i,j] is the expression level of gene i of sample j. The matrix has dimension (nt2,ns).
nt	Number of genes for A.
nt2	Number of genes for B.
ns	Number of samples.

nodiag	When A and B are the same, log likelihood ratio between alternative and null hypotheses gives infinity. To avoid its contamination in the conversion from log likelihood ratios into probabilities, users need to arrange data accordingly, when $\{A\}$ and $\{B\}$ are the same or when $\{A\}$ is a subset of $\{B\}$. The top submatrix of B's expression data must be identical with A, and nodiag must be set to 1, True, or TRUE. Otherwise, in the default configuration, $\{A\}$ and $\{B\}$ should not have any intersection and nodiag = 0, False, or FALSE.
memlimit memlimit	The approximate memory usage limit in bytes for the library. For datasets require a larger memory, calculation will be split into smaller chunks. If the memory limit is smaller than minimum required, calculation can fail with an error message. Use the default value 0 to indicate unlimited memory usage. R version does not provide this limit and only uses unlimited memory, because R itself uses extra memory.
fp ans[‘p’] ans	Output matrix of inferred probability of correlation $A \cdots B$ versus no correlation $A \quad B$, in file path, numpy.ndarray, or matrix format. Element [i,j] is the probability of gene i being correlated with gene j. The matrix has dimension (nt,nt2).

- Perform 5 tests that may assist the inference of regulation $A \to B$. Input of expression data of A, B and best eQTL data for A as $E(A)$ are accepted. Posterior probabilities of 5 tests are outputed separately. Methodology can be found in [2]. Bayesian inferences from log likelihood ratios to probabilities are drawn for each A and each test.
  pijs_gassist fg ft ft2 nt nt2 ns fp1 fp2 fp3 fp4 fp5 na nodiag memlimit
  ans=l.pijs_gassist(dg,dt,dt2,na=None,nodiag=False,memlimit=0)
  ans=findr.pijs_gassist(dg,dt,dt2,na=NULL,nodiag=FALSE)

fg dg dg	Input matrix of best eQTL genotype data $E(A)$, each row of which is the best eQTL of the corresponding row of ft,dt. Data is in file path, numpy.ndarray, or matrix format. Element [i,j] is the genotype value of the best eQTL of gene i of sample j, and should be among values $0, 1, \ldots, na$. The matrix has dimension (nt,ns).
ft dt dt	Input matrix of expression levels of A, in file path, numpy.ndarray, or matrix format. Element [i,j] is the expression level of gene i of sample j. The matrix has dimension (nt,ns).
ft2 dt2 dt2	Input matrix of expression levels of B, in file path, numpy.ndarray, or matrix format. Element [i,j] is the expression level of gene i of sample j. The matrix has dimension (nt2,ns).
nt	Number of genes for A.
nt2	Number of genes for B.
ns	Number of samples.
na	Number of alleles for the species considered. This constrains every genotype data to be among $0, 1, \ldots, na$. If unspecified (0, None, or NULL), na is automatically determined as the maximum value of fg,dg + 1.
nodiag	When A and B are the same, log likelihood ratio between alternative and null hypotheses can give infinity. To avoid its contamination in the conversion from log likelihood ratios into probabilities, users need to arrange data accordingly, when $\{A\}$ and $\{B\}$ are the same or when $\{A\}$ is a subset of $\{B\}$. The top submatrix of B's expression data must be identical with A, and nodiag must be set to 1, True, or TRUE. Otherwise, in the default configuration, $\{A\}$ and $\{B\}$ should not have any intersection and nodiag = 0, False, or FALSE.

memlimit memlimit	The approximate memory usage limit in bytes for the library. For datasets require a larger memory, calculation will be split into smaller chunks. If the memory limit is smaller than minimum required, calculation can fail with an error message. Use the default value 0 to indicate unlimited memory usage. R version does not provide this limit and only uses unlimited memory, because R itself uses extra memory.
fp1 ans['p1'] ans$p1	Output vector of inferred probability of test 1, $E(A) \to A$ (alternative) versus $E(A) \quad A$ (null), in file path, numpy.ndarray, or array format. Element [i] is the probability of best eQTL of gene i regulates gene i. The vector has dimension (nt). For nodiag=0, False, FALSE, because the function expects significant eQTLs, p1 always return 1. For nodiag=1, True, TRUE, uses diagonal elements of p2. Consider replacing p1 with your own (1-FDR) from eQTL discovery.
fp2 ans['p2'] ans$p2	Output matrix of inferred probability of test 2, $E(A) \to A \cdots B$ with $E(A) \to B$ (alternative) versus $E(A) \to A \leftarrow B$ (null), in file path, numpy.ndarray, or matrix format. Element [i,j] is the probability of alternative hypothesis for A = gene i and B = gene j. The matrix has dimension (nt,nt2).
fp3 ans['p3'] ans$p3	Output matrix of inferred probability of test 3, $E(A) \to A \to B$ (null) versus $E(A) \to A \cdots B$ with $E(A) \to B$ (alternative), in file path, numpy.ndarray, or matrix format. Element [i,j] is the probability of null hypothesis for A = gene i and B = gene j. The matrix has dimension (nt,nt2).
fp4 ans['p4'] ans$p4	Output matrix of inferred probability of test 4, $B \leftarrow E(A) \to A$ with $A \cdots B$ (alternative) versus $E(A) \to A \quad B$ (null), in file path, numpy.ndarray, or matrix format. Element [i,j] is the probability of alternative hypothesis for A = gene i and B = gene j. The matrix has dimension (nt,nt2).
fp5 ans['p5'] ans$p5	Output matrix of inferred probability of test 5, $B \leftarrow E(A) \to A$ with $A \cdots B$ (alternative) versus $B \leftarrow E(A) \to A$ (null), in file path, numpy.ndarray, or matrix format. Element [i,j] is the probability of alternative hypothesis for A = gene i and B = gene j. The matrix has dimension (nt,nt2).

- Perform 5 tests and combine them so as to obtain a single probability for the regulation $A \to B$. The combination either follows [2], or traditional causal inference test ([5]). Input of expression data of A, B and best eQTL data for A as $E(A)$ are accepted. The combination in [2] is recommended, which is $(p_2p_5 + p_4)/2$.

  (Recommended:) For combination in [2] with Bayesian conversion from log likelihood ratios to probabilities drawn for each A and each test:
  pij_gassist fg ft ft2 nt nt2 ns fp na nodiag memlimit
  ans=l.pij_gassist(dg,dt,dt2,na=None,nodiag=False,memlimit=0)
  ans=findr.pij_gassist(dg,dt,dt2,na=NULL,nodiag=FALSE)

  For combination in [5] with Bayesian conversion from log likelihood ratios to probabilities drawn for each A and each test[1]:
  pij_gassist_trad fg ft ft2 nt nt2 ns fp na nodiag memlimit
  ans=l.pij_gassist_trad(dg,dt,dt2,na=None,nodiag=False,memlimit=0)
  ans=findr.pij_gassist_trad(dg,dt,dt2,na=NULL,nodiag=FALSE)

fg dg dg	Input matrix of best eQTL genotype data $E(A)$, each row of which is the best eQTL of the corresponding row of ft,dt. Data is in file path, numpy.ndarray, or matrix format. Element [i,j] is the genotype value of the best eQTL of gene i of sample j, and should be among values $0, 1, \ldots, na$. The matrix has dimension (nt,ns).

[1] Note: this is not intended as a loyal reimplementation of Trigger. Instead, it simply performs the tests suggested in Trigger, but all preprocessing, postprocessing, and exact hypotheses can be different. A significant overlap of the top predictions of this method and Trigger has been observed in existing studies.

ft dt dt	Input matrix of expression levels of A, in file path, numpy.ndarray, or matrix format. Element [i,j] is the expression level of gene i of sample j. The matrix has dimension (nt,ns).
ft2 dt2 dt2	Input matrix of expression levels of B, in file path, numpy.ndarray, or matrix format. Element [i,j] is the expression level of gene i of sample j. The matrix has dimension (nt2,ns).
nt	Number of genes for A.
nt2	Number of genes for B.
ns	Number of samples.
na	Number of alleles for the species considered. This constrains every genotype data to be among $0, 1, \ldots, na$. If unspecified (0, None, or NULL), na is automatically determined as the maximum value of fg,dg + 1.
nodiag	When A and B are the same, log likelihood ratio between alternative and null hypotheses can give infinity. To avoid its contamination in the conversion from log likelihood ratios into probabilities, users need to arrange data accordingly, when $\{A\}$ and $\{B\}$ are the same or when $\{A\}$ is a subset of $\{B\}$. The top submatrix of B's expression data must be identical with A, and nodiag must be set to 1, True, or TRUE. Otherwise, in the default configuration, $\{A\}$ and $\{B\}$ should not have any intersection and nodiag = 0, False, or FALSE.
memlimit memlimit	The approximate memory usage limit in bytes for the library. For datasets require a larger memory, calculation will be split into smaller chunks. If the memory limit is smaller than minimum required, calculation can fail with an error message. Use the default value 0 to indicate unlimited memory usage. R version does not provide this limit and only uses unlimited memory, because R itself uses extra memory.
fp ans['p'] ans	Output matrix of inferred probability after combination of the five tests, in file path, numpy.ndarray, or matrix format. Element [i,j] is the probability of alternative hypothesis for A = gene i and B = gene j. The matrix has dimension (nt,nt2).

4.3 Reconstruction of directed acyclic graphs

This section provides efficient reconstruction of directed acyclic graphs from prior information of edge significance. As an example, the edge significance may come from the output of the inference of pairwise regulations, e.g. pij_gassist or pij_rank.

The exposed functions for reconstruction of directed acyclic graphs are listed below:

- Reconstruction of a single directed acyclic graph from the given edge significance prior by adding the most significant edge one at a time, whilst avoiding cycles. Methodology can be found in [7, 3].
  netr_one_greedy fprior nt fnet namax nimax nomax
  ans=l.netr.one_greedy(prior,namax=None,nimax=None,nomax=None)
  ans=findr.netr_one_greedy(prior,namax=NULL,nimax=NULL,nomax=NULL)

fprior prior prior	Input matrix of significance levels of all edges, in file path, numpy.ndarray, or matrix format. Element [i,j] is the significance level of edge i to j. The matrix has dimension (nt,nt).
nt	Number of nodes to reconstruct network for.
namax	Maximum total number of edges in the reconstructed network. Defaults (0 for binary interface) to unlimited, i.e. nt*(nt-1)/2.
nimax	Maximum number of incoming edges for each node in the reconstructed network. Defaults (0 for binary interface) to unlimited.
nomax	Maximum number of outgoing edges for each node in the reconstructed network. Defaults (0 for binary interface) to unlimited.

fnet ans['net'] ans	Output boolean matrix of reconstructed network, in file path (8-bit integer for raw format), numpy.ndarray(dtype=bool), or matrix format. Element [i,j]=1, True, TRUE if an edge exists from node i to j on the reconstructed network, and 0, False, FALSE if not. The matrix has dimension (nt,nt).

5 Examples

Part of GEUVADIS Consortium dataset is provided within the binary and python2 interfaces, as well as the R package. Usage of examples is provided in every distribution as below:

- **Binary interface**: see file EXAMPLES.
- **Python2 interface**: see module findr.examples.
- **R package**: see documentation of every function or Findr package.

6 Frequently asked questions

1. I have GCC on my Mac but when compiling Findr still asks me to download GCC.

   On Mac, Apple installs its own C compiler which tries to pretend to be GCC, although many functionalities of GCC are lacked in Apple's compiler. Findr needs some of these functionalities (such as OpenMP). You can download the source code of GCC from https://gcc.gnu.org. Because of the complications in building GCC, some Mac users prefer to download unofficial binary copies of GCC from third parties, such as Homebrew. The unofficial binary GCC may be installed under a name other than 'gcc'. Under such circumstances, consult the question 'How do I change C compiler name?'

2. How do I change C compiler name?

   For Findr library or binary interface, open Makefile with any text editor. Find the lines 'CC=gcc' and 'LD=gcc', and change gcc into your C compiler name. After that, run 'make distclean' before jumping back at the Build phase of installation in Section 2.1. For Findr R package, go to src/lib and perform the same operation. For python2 interface, no C code is included so no change is needed.

3. Can I use Findr if I don't have admin rights?

   Yes. Suppose you would like to install Findr to /path/to/install. You can install bothFindr library and binary as the following:

   ```
   export PREFIX=/path/to/install
   make -e
   make -e install
   ```

   After that, you should also include /path/to/install in the include, library and binary paths of the system. This can be done by adding the following lines in the .bashrc file in your home folder (or accordingly if you use other shells):

   ```
   export PATH=/path/to/install/bin:$PATH
   export LD_LIBRARY_PATH=/path/to/install/lib:$LD_LIBRARY_PATH
   export LIBRARY_PATH=/path/to/install/lib:$LIBRARY_PATH
   export CPATH=/path/to/install/include:$CPATH
   ```

4. Does Findr support Windows?

   Findr supports Windows in a number of ways. The computation code itself is platform independent but installation of GSL can be complicated on Windows. Findr compiles and runs smoothly on *nix port platforms on Windows, such as Cygwin or MSYS2. In previous versions, we released MSYS2 based binary distributions of Findr on Windows.

   Ever since Windows released support for Ubuntu Bash recently, we have decided to drop Windows native support. Findr and all its interfaces should integrate better with Bash on Windows. (https://msdn.microsoft.com/commandline/wsl)

References

[1] Lingfei Wang and Tom Michoel. Comparable variable selection with Lasso. *arXiv preprint arXiv:1701.07011*, 2017.

[2] Lingfei Wang and Tom Michoel. Efficient causal inference with hidden confounders from genome-transcriptome variation data. *arXiv preprint arXiv:1610.03123*, 2016.

[3] Lingfei Wang and Tom Michoel. Scalable causal gene network inference with prior node ordering and posterior FDR control. *in preparation*, 2017.

[4] Joshua Millstein, Bin Zhang, Jun Zhu, and Eric E. Schadt. Disentangling molecular relationships with a causal inference test. *BMC Genetics*, 10(1):1–15, 2009.

[5] Lin S. Chen, Dipen P. Sangurdekar, and John D. Storey. *trigger: Transcriptional Regulatory Inference from Genetics of Gene ExpRession*, 2007. R package version 1.16.0.

[6] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):1–10, 09 2010.

[7] Bernhard Haeupler, Telikepalli Kavitha, Rogers Mathew, Siddhartha Sen, and Robert E. Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Trans. Algorithms*, 8(1):3:1–3:33, January 2012.