ParallelPC: An R Package for Efficient Causal Exploration in Genomic Data

Thuc Duy Le[1(✉)], Taosheng Xu[2], Lin Liu[1], Hu Shu[1], Tao Hoang[1], and Jiuyong Li[1]

[1] University of South Australia, Mawson Lakes, SA 5095, Australia
{Thuc.Le,Lin.Liu,Jiuyong.Li}@unisa.edu.au, hushu@iie.ac.cn, hoatn002@mymail.unisa.edu.au

[2] Institute of Intelligent Machines, Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China
xutaosheng@aliyun.com

Abstract. Discovering causal relationships from genomic data is the ultimate goal in gene regulation research. Constraint based causal exploration algorithms, such as PC, FCI, RFCI, PC-simple, IDA and Joint-IDA have achieved significant progress and have many applications. However, their applications in bioinformatics are still limited due to their high computational complexity. In this paper, we present an R package, *ParallelPC*, that includes the parallelised versions of these causal exploration algorithms and 12 different conditional independence tests for each. The parallelised algorithms help speed up the procedure of experimenting large biological datasets and reduce the memory used when running the algorithms. Our experiment results on a real gene expression dataset show that using the parallelised algorithms it is now practical to explore causal relationships in high dimensional datasets with thousands of variables in a personal multicore computer. We present some typical applications in bioinformatics using different algorithms in *ParallelPC*. *ParallelPC* is available in CRAN repository at https://cran.r-project.org/web/packages/ParallelPC/index.html.

Keywords: Causality discovery · Bayesian networks
Parallel computing · Constraint-based methods

1 Introduction

Inferring causal relationships between variables is an ultimate goal of many research areas, e.g. investigating the causes of cancer, finding the factors affecting life expectancy. Therefore, it is important to develop tools for causal exploration from real world datasets.

One of the most advanced theories with widespread recognition in discovering causality is the Causal Bayesian Network (CBN), [1]. In this framework, causal relationships are represented with a Directed Acyclic Graph (DAG). There are two main approaches for learning the DAG from data: the search and score approach, and the constraint based approach. While the search and score approach raises an NP-hard problem, the complexity of the constraint based approach is exponential to the number of variables. Constraint based approach for causality discovery has been advanced in the last decade and has been shown to be useful in some real world applications. The approach includes causal structure learning methods, e.g. PC [2], FCI and RFCI [3], causal inference methods, e.g. IDA [4] and Joint-IDA [5], and local causal structure learning such as PC-Simple [6,7]. However, the high computational complexity has hindered the applications of causal discovery approaches to high dimensional datasets, e.g. gene expression datasets where the number of genes (variables) is large and the number of samples is normally small.

In [8], we presented a method that is based on parallel computing technique to speed up the PC algorithm. Here in the *ParallelPC* package, we parallelise a family of causal structure learning and causal inference methods, including PC, FCI, RFCI, PC-simple, IDA, and Joint-IDA. We also collate 12 different conditional independence (CI) tests that can be used in these algorithms. The algorithms in this package return the same results as those in the *pcalg* package [9], but the runtime is much lower depending on the number of cores CPU specified by users. Our experiment results show that with the *ParallelPC* package it is now practical to apply those methods to genomic datasets in a modern personal computer.

2 Contraint Based Algorithms and Their Parallelised Versions

We paralellised the following causal discovery and inference algorithms.

– PC, [2]. The PC algorithm is the state of the art method in constraint based approach for learning causal structures from data. It has two main steps. In the first step, it learns from data a skeleton graph, which contains only undirected edges. In the second step, it orients the undirected edges to form an equivalence class of DAGs. In the skeleton learning step, the PC algorithm starts with the fully connected network and uses the CI tests to decide if an edge is removed or retained. The stable version of the PC algorithm (the Stable-PC algorithm, [10]) updates the graph at the end of each level (the size of the conditioning set) of the algorithm rather than after each CI test. Stable-PC limits the problem of the PC algorithm, which is dependent on the order of the CI tests. It is not possible to parallelise the Stable-PC algorithm globally, as the CI tests across different levels in the Stable-PC algorithm are dependent to one another. In [8], we proposed the Parallel-PC algorithm which parallelised the CI tests inside each level of the Stable-PC algorithm. The Parallel-PC algorithm is more efficient and returns the same results as that of the Stable-PC algorithm (see Fig. 1).

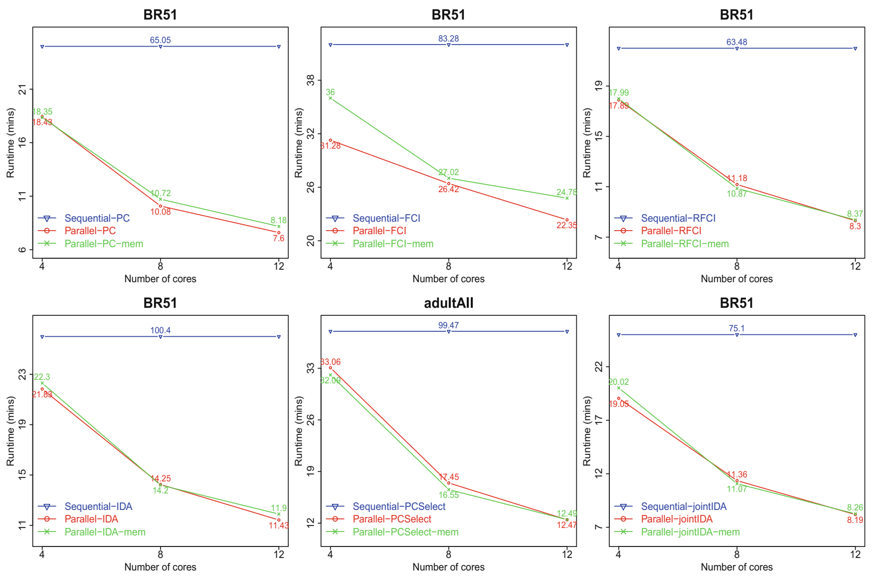

Fig. 1. Runtime of the sequential and parallelised versions (with and without the memory efficient option) of PC, FCI, RFCI, IDA, PC-simple, and Joint-IDA

– FCI, [3]. FCI is designed for learning the causal structure that takes latent variables into consideration. In real world datasets, there are often unmeasured variables and they will affect the learnt causal structure. FCI was implemented in the *pcalg* package and it uses PC algorithm as the first step. The skeleton of the causal structure learnt by PC algorithm will be refined by performing more CI test. Therefore, FCI is not efficient for large datasets.
– RFCI, [3]. RFCI is an improvement of the FCI algorithm to speed up the running time when the underlying graph is sparse. However, our experiment results show that it is still impractical for high dimensional datasets.
– PC-simple, [6]. PC-simple is a local causal discovery algorithm to search for parents and children of the target variable. In dense datasets where the target variable has large number of causes and effects, the algorithm is not efficient. We utilise the idea of taking order-independent approach [8] on the local structure learning problem to parallelise the PC-simple algorithm.
– IDA, [4]. IDA is a causal inference method which infers the causal effect that a variable has on another variable. It firstly learns the causal structure from data, and then based on the learnt causal structure, it estimates the causal effect between a cause node and an effect node by adjusting the effects of the parents of the cause. Learning the causal structure is time consuming. Therefore, IDA will be efficient when the causal structure learning step is improved. Figure 1 shows that our parallelised version of the IDA improves the efficiency of the IDA algorithm significantly.

– Joint-IDA, [5]. Joint-IDA estimates the effect of the target variable when jointly intervening a group of variables. Similar to IDA, Joint-IDA learns the causal structure from data and the effects of intervening multiple variables are estimated.

To illustrate the effectiveness of the parallelised algorithms, we apply the sequential and parallelised versions of PC, FCI, RFCI, IDA and Joint-IDA algorithms to a breast cancer gene expression dataset. The dataset includes 50 expression samples with 92 microRNAs (a class of gene regulators) and 1500 messenger RNAs that was used to infer the relationships between miRNAs and mRNAs downloaded from [11]. As PC-simple (PC-Select) is efficient in small datasets, we use the Adult dataset from UCI Machine Learning Repository with 48842 samples. We use the binary discretised version from [12,13] and select 100 binary variables for the experiment with PC-simple. We run all the experiments on a Linux server with 2.7 GB memory and 2.6 GHz per core CPU.

As shown in Fig. 1, the parallelised versions of the algorithms are much more efficient than the sequential versions as expected, while they are still generating the same results.

The parallelised algorithms also detect the free memory of the running computer to estimate the number of CI tests that will be distributed evenly to the cores. This step is to ensure that each core of the computer will not hold off a big amount of memory while waiting for the synchronisation step. The memory-efficient procedure may consume a little bit more time compared to the original parallel version. However, this option is recommended for computers with limited memory resources or for big datasets.

3 Conditional Independence Tests for Constraint Based Methods

It is shown that different CI tests may lead to different results for a particular constraint based algorithm, and a CI test may be suitable for a certain type of datasets. In this package, we collate 12 CI tests in the *pcalg* [9] and *bnlearn* [14] packages to provide options for function calls of the constraint-based methods. These CI tests can be used separately for the purpose of testing (conditional) dependency between variables. They can also be used within the constraint based algorithms (both sequential and parallelised algorithms) in the *ParallelPC* package. The following codes show an example of running the FCI algorithm using the sequential version in the *pcalg* package, the parallelised versions with or without the memory efficient option, and using a different CI test (mutual information) rather than the Gaussian CI test.

```
## Using the FCI-stable algorithm in the pcalg package
library(pcalg)
data("gmG")
p<-ncol(gmG$x)
```

```
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
fci_stable(suffStat, indepTest=gaussCItest, p=p,
skel.method="stable", alpha=0.01)
## Using fci_parallel without the memory efficient option
fci_parallel(suffStat, indepTest=gaussCItest, p=p,
skel.method="parallel",
alpha=0.01, num.cores=2)
## Using fci_parallel with the memory efficient option
fci_parallel(suffStat, indepTest=gaussCItest, p=p,
skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)
## Using fci_parallel with mutual information test
fci_parallel(gmG$x, indepTest=mig, p=p,
skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)
```

4 Finding a Group of Genes Directly Connected to a Gene of Interest

Given a gene of interest, we may want to see all genes that have direct causal relationship with the interest gene. The following workflow shows the convenient approach to achieve the analysis target by using the *pcSelect_parallel()* function in the *ParallelPC* package. We take the TP53 gene as an example gene of interest. We aim to infer the parent and children genes of TP53 within the top 40 differentially expression genes in the TCGA BRCA dataset.

```

## Select a gene of interest
index3=which(rownames(Tumor_Exp)=="BRCA1")
BRCA1_exp=Tumor_Exp[index3 ,]

## Use pcSelect() function to find the set of parents and
## children of the gene of interest.
pcSelect_Result<- pcSelect(BRCA1_exp,BRCA_mRNA,alpha=0.01)

##Select the causal effect features
index4=which(pcSelect_Result$G==TRUE)
pcSelect_Result_1=data.frame(
"mRNA"=names(pcSelect_Result$G)[index4]
,"zMin"=pcSelect_Result$zMin[index4])

##       mRNA      zMin
##1  PAFAH1B3  3.735917
##2     FXYD1  2.788022
##3      NEK2  5.868730
##4   ADAMTS5  3.216070
```

5 Predicting miRNA Targets Using a Causal Inference Method

The regulatory relationship between an miRNA and an mRNA means that a change in the expression level of the miRNA results in the change in the expression level of the mRNA. Causal inference methods allow us to estimate the change of a gene when manipulating another and therefore applicable for identifying the causal effects that the miRNAs have on the mRNAs [15–17]. The estimated causal effect is different from the correlation coefficient as in traditional correlation-based methods. In this section, we present the usage of the *ParallelPC* package in predicting the targets of a miRNA with a causal inference method called IDA.

5.1 Obtaining a Matched miRNA, mRNA Gene Expression Dataset

Firstly we retrieve the mRNA and miRNA expression data from TCGA BRCA dataset.

```
rm(list = ls())
####Retrieve TCGA mRNA and miRNA expression dataset
library("RTCGA.mRNA")
library("RTCGA.miRNASeq")
#### Extract the mRNA expression data of breast cancer
data(BRCA.mRNA)
mRNA=t(as.matrix(BRCA.mRNA[,-1]))
colnames(mRNA)=BRCA.mRNA[,1]
#### data imputation for the missing measurements
library("impute")
mRNA=impute.knn(mRNA)$data
###Split the normal and tumor samples
index=which(as.numeric(substr(colnames(mRNA),14,15))>9)
Normal_Exp=mRNA[,index]
Tumor_Exp=mRNA[,-index]
### Remove the duplicated tumor samples
index1=which(as.numeric(substr(colnames(Tumor_Exp),14,15))>1)
Tumor_Exp=Tumor_Exp[,-index1]
colnames(Tumor_Exp)=substr(colnames(Tumor_Exp),1,12)

#### Extract the miRNA expression data of breast cancer
data(BRCA.miRNASeq)
BRCA.miRNASeq=
BRCA.miRNASeq[seq(2, nrow(BRCA.miRNASeq), 3),-c(1,2)]
miRNASeq=apply(as.matrix(BRCA.miRNASeq),1,as.numeric)
rownames(miRNASeq)=colnames(BRCA.miRNASeq)
###Split the normal and tumor samples
index3 =which(as.numeric(substr(colnames(miRNASeq),14,15))>9)
Normal_miRNA=miRNASeq[,index3]
Tumor_miRNA=miRNASeq[,-index3]
### Remove the duplicated samples
```

```
index4=
which(as.numeric(substr(colnames(Tumor_miRNA),14,15))>1)
Tumor_miRNA=Tumor_miRNA[,-index4]
colnames(Tumor_miRNA)=substr(colnames(Tumor_miRNA),1,12)
```

5.2 Finding the Differentially Expressed miRNAs and mRNAs

```
library(limma)
c1=ncol(Normal_Exp)
c2=ncol(Tumor_Exp)
mR=cbind(Normal_Exp, Tumor_Exp)
design1=cbind(Normal=c(rep(1,c1), rep(0,c2)),
Cancer=c(rep(0,c1), rep(1,c2)))
##In order to return the index of features ,
##set the same name for two feature
rownames(mR)[1]="repeat"
rownames(mR)[2]="repeat"
mRfit=lmFit(mR, design1)
contrast.matrix=makeContrasts(NormalvCancer=Normal - Cancer ,
levels=design1)
mRfit1=contrasts.fit(mRfit , contrast.matrix)
mRfit1=eBayes(mRfit1)
##Choose the top 100 differentially expression genes
##for experiment analysis
mRresults=
topTable(mRfit1 , number= 100, sort.by="p", adjust="BH")
index2= as.numeric(row.names(mRresults))
##Extract the experiment dataset for downstream analysis
BRCA_mRNA_100=t(Tumor_Exp[index2 ,])

##Extract the most different expression miRNAs
c1=ncol(Normal_miRNA)
c2=ncol(Tumor_miRNA)
miRNA=cbind(Normal_miRNA, Tumor_miRNA)
design2=cbind(Normal=c(rep(1,c1), rep(0,c2)),
Cancer=c(rep(0,c1), rep(1,c2)))
###In order to return the index of features,
###set the same name for two feature
rownames(miRNA)[1]="repeat"
rownames(miRNA)[2]="repeat"
miRNA <- voom(miRNA, design2, plot=TRUE)
mRfit=lmFit(miRNA, design2)
contrast.matrix=makeContrasts(NormalvCancer=Normal - Cancer,
levels=design2)
mRfit2=contrasts.fit(mRfit, contrast.matrix)
mRfit2=eBayes(mRfit2)
##Choose the top 10 differentially expression miRNAs
##for experiment analysis
```

```
mRresults=
topTable(mRfit2, number= 10, sort.by="p", adjust="BH")
index5 = as.numeric(row.names(mRresults))
##Get the final experiment dataset.
BRCA_miRNA_10=t(Tumor_miRNA[index5,])
```

5.3 Extract the Matched Samples with mRNA and miRNA

```
mRNA_samples=rownames(BRCA_mRNA_100)
miRNA_samples=rownames(BRCA_miRNA_10)
intersect_samples=intersect(mRNA_samples,miRNA_samples)
```

```
index6 =match(intersect_samples,mRNA_samples)
index7=match(intersect_samples,miRNA_samples)
BRCA_mRNA_100_matched=BRCA_mRNA_100[index6,]
BRCA_miRNA_10_matched=BRCA_miRNA_10[index7,]
##Test all samples are matched or not
all(rownames(BRCA_mRNA_100_matched)==
rownames(BRCA_miRNA_10_matched))

BRCA_matched_miRNA10_mRNA100=cbind(BRCA_miRNA_10_matched,
BRCA_mRNA_100_matched)
write.csv(BRCA_matched_miRNA10_mRNA100,
file = "BRCA_matched_miRNA10_mRNA100.csv",
row.names = FALSE)
```

5.4 Applying Parallel IDA for Inferring miRNA-mRNA Causal Effects Without Using Target Binding Information

```
library("ParallelPC")
library("pcalg")
library("parallel")
library("miRLAB")
miRNAFrom=1
miRNATo=10
mRNAFrom=11
mRNATo=110
IDAResult_noTargetBinding =IDA_parallel(
"BRCA_matched_miRNA10_mRNA100.csv",
miRNAFrom:miRNATo,mRNAFrom:mRNATo,"parallel",0.01, 2, TRUE)
##Extract the top 20 targets of hsa-mir-139 as an example
library("miRLAB")
miRTop20_1=bRank(IDAResult_noTargetBinding, 2,20, TRUE)
##         miRNA       mRNA    Causal effects
##51  hsa-mir-139     PDE2A    0.4176818
```

```
##15 hsa-mir-139      BTNL9   0.3658065
##4  hsa-mir-139        CA4   0.3655496
##98 hsa-mir-139     AQP7P2   0.3512459
##79 hsa-mir-139      LYVE1   0.3499394
##7  hsa-mir-139    CD300LG   0.3488489
##31 hsa-mir-139      AVPR2   0.3420044
##68 hsa-mir-139       NPR1   0.3372393
##91 hsa-mir-139   MAP1LC3C   0.3324548
##12 hsa-mir-139      ATOH8   0.3232092
##78 hsa-mir-139     KCNIP2   0.3205980
##87 hsa-mir-139       PCK1   0.3198771
##44 hsa-mir-139     ACVR1C   0.3163645
##21 hsa-mir-139   HSD17B13   0.3156995
##13 hsa-mir-139  LOC387911   0.3138352
##33 hsa-mir-139       FIGF   0.3131615
##20 hsa-mir-139    C1QTNF9   0.3109739
##57 hsa-mir-139       TNMD   0.3081662
##28 hsa-mir-139       SDPR   0.3051225
##37 hsa-mir-139      ITIH5   0.3050095
```

5.5 Applying IDA for Inferring miRNA-mRNA Causal Effects with Target Binding Information

We use the IDA() function from *miRLAB* package [18] to infer the miRNA-mRNA causal effects with the TargetScan 7 as the target binding information. The TargetScan7.csv file can be downloaded from (`nugget.unisa.edu.au/Thuc/TargetScan7.csv`).

```
library("miRLAB")
IDAResult_TargetBinding =
IDA("BRCA_matched_miRNA10_mRNA100.csv",
miRNAFrom:miRNATo, mRNAFrom:mRNATo,
targetbinding = "TargetScan7.csv")
```

5.6 Creating an Ensemble Method by Combining IDA and Lasso

```
### Ensemble method
library(miRLAB)
IDA_Ensemble =
IDA_parallel("BRCA_matched_miRNA10_mRNA100.csv",
cause = miRNAFrom:miRNATo,
effect = mRNAFrom:mRNATo,
"parallel",0.01, 2, TRUE)

Lasso_Ensemble = Lasso("BRCA_matched_miRNA10_mRNA100.csv",
cause=miRNAFrom:miRNATo,
effect=mRNAFrom:mRNATo)
Borda_Ensemble = Borda(list(Lasso_Ensemble, IDA_Ensemble))
miRTop20_2= bRank(Borda_Ensemble,2,20,TRUE )

## Extract the top 20 targets of hsa-mir-139 as an example
##         miRNA      mRNA  Ranking Score
##51 hsa-mir-139     PDE2A  100.000000
##98 hsa-mir-139    AQP7P2   28.571429
##4  hsa-mir-139       CA4   28.571429
##7  hsa-mir-139   CD300LG   25.000000
##15 hsa-mir-139     BTNL9   18.181818
##79 hsa-mir-139     LYVE1   15.384615
##68 hsa-mir-139      NPR1   15.384615
##78 hsa-mir-139    KCNIP2   11.111111
##44 hsa-mir-139    ACVR1C   10.526316
##87 hsa-mir-139      PCK1    9.090909
##91 hsa-mir-139  MAP1LC3C    8.333333
##21 hsa-mir-139  HSD17B13    8.000000
##31 hsa-mir-139     AVPR2    7.407407
##57 hsa-mir-139      TNMD    6.250000
##6  hsa-mir-139     GLYAT    6.060606
##33 hsa-mir-139      FIGF    5.714286
##19 hsa-mir-139      AQP7    5.128205
##13 hsa-mir-139 LOC387911    5.128205
##61 hsa-mir-139    GPR146    5.000000
##12 hsa-mir-139     ATOH8    5.000000
```

6 Identifying Double Intervention Effects from Expression Data

While IDA estimates the causal effect of one variable on the other, joint_IDA estimate the joint causal effect of multiple variables on the other. In this scenario, we show the example of calculating the joint causal effect of 2 miRNAs on a target gene. The joint causal effect here is similar to the effect of knocking down the two miRNAs at the same time.

6.1 Identify Differentially Expressed Genes for the TCGA Breast Cancer Dataset

We extract the top 40 differentially expressed mRNAs and 10 differentially expressed miRNAs for the analysis.

```

BRCA_mRNA_40_matched=BRCA_mRNA_100_matched[,1:40]
BRCA_matched_miRNA10_mRNA40=cbind(BRCA_miRNA_10_matched,
BRCA_mRNA_40_matched)
write.csv(BRCA_matched_miRNA10_mRNA40,
file = "BRCA_matched_miRNA10_mRNA40.csv",
row.names = FALSE)
```

6.2 Identify Join-Effect of 2 miRNAs on a Gene

We find the joint causal effect of *hsa-mir-592* and *hsa-mir-139* on *RDH5*.

```

causal_coefficient=
jointIDA_parallel("BRCA_matched_miRNA10_mRNA40.csv", 1:2,40,
pcmethod="parallel", 0.01, 2, technique="RRC")
##hsa.mir.592 -0.1999714
##hsa.mir.139  0.2391669

#Let X1 be the average expression level of  hsa-miR-592
#and X2 be the average expression level of hsa-miR-139
X1=colMeans(BRCA_miRNA_10_matched)[1]# hsa_mir_592
X2=colMeans(BRCA_miRNA_10_matched)[2]# hsa_mir_139

#The joint causal effect of the two miRNAs on RDH5 is:
Joint_effect = -X1*( -0.1999714)-X2*0.2391669
##The result is -10.17413.
```

7 Conclusion

In this paper, we present a software package, *ParallelPC*, for efficient causal exploration using genomic data. We present some use cases of the package including (i) inferring gene regulatory networks, (ii) finding a set of genes having causal relationship with a gene of interest, (iii) predicting miRNA targets with causal inference methods, and (iv) identifying the joint causal effect of multiple miRNAs on a given gene. The package will offer a set of useful causal exploration tools for novel applications.

Funding
Thuc Duy Le was supported by NHMRC [grant ID 1123042]. This work has been partly supported by ARC Discovery Project DP170101306.

References

1. Pearl, J.: Causality. Cambridge University Press, Cambridge (2009)
2. Spirtes, P., Glymour, C.N., Scheines, R.: Causation, Prediction, and Search, vol. 81. MIT Press, Cambridge (2000)
3. Colombo, D., Maathuis, M.H., Kalisch, M., Richardson, T.S., et al.: Learning high-dimensional directed acyclic graphs with latent and selection variables. Ann. Stat. **40**(1), 294–321 (2012)

4. Maathuis, M.H., Kalisch, M., Bühlmann, P., et al.: Estimating high-dimensional intervention effects from observational data. Ann. Stat. **37**(6A), 3133–3164 (2009)
5. Nandy, P., Maathuis, M.H., Richardson, T.S.: Estimating the effect of joint interventions from observational data in sparse high-dimensional settings. Ann. Stat. **45**(2), 647–674 (2017)
6. Bühlmann, P., Kalisch, M., Maathuis, M.H.: Variable selection in high-dimensional linear models: partially faithful distributions and the PC-simple algorithm. Biometrika **97**(2), 261–278 (2010)
7. Li, J., Liu, L., Le, T.D.: Practical Approaches to Causal Relationship Exploration. SECE. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14433-7
8. Le, T.D., Hoang, T., Li, J., Liu, L., Liu, H., Hu, S.: A fast PC algorithm for high dimensional causal discovery with multi-core PCs. IEEE/ACM Trans. Comput. Biol. Bioinform. (2016)
9. Kalisch, M., Mächler, M., Colombo, D., Maathuis, M.H., Bühlmann, P.: Causal inference using graphical models with the R package pcalg. J. Stat. Softw. **47**(11), 1–26 (2012)
10. Colombo, D., Maathuis, M.H.: Order-independent constraint-based causal structure learning. J. Mach. Learn. Res. **15**(1), 3741–3782 (2014)
11. Le, T.D., Liu, L., Zhang, J., Liu, B., Li, J.: From miRNA regulation to miRNA-TF co-regulation: computational approaches and challenges. Brief. Bioinform. **16**(3), 475–496 (2015)
12. Li, J., Le, T.D., Liu, L., Liu, J., Jin, Z., Sun, B.: Mining causal association rules. In: 2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW), pp. 114–123 (2013)
13. Li, J., Le, T.D., Liu, L., Liu, J., Jin, Z., Sun, B., Ma, S.: From observational studies to causal rule mining. ACM Trans. Intell. Syst. Technol. **7**(2), 14 (2016)
14. Scutari, M.: Learning Bayesian networks with the bnlearn R package. arXiv preprint arXiv:0908.3817 (2009)
15. Le, T.D., et al.: Inferring microRNA–mRNA causal regulatory relationships from expression data. Bioinformatics **29**(6), 765–771 (2013). https://doi.org/10.1093/bioinformatics/btt048
16. Zhang, J., et al.: Inferring condition-specific miRNA activity from matched miRNA and mRNA expression data. Bioinformatics **30**(21), 3070–3077 (2014)
17. Zhang, J., et al.: Identifying direct miRNA–mRNA causal regulatory relationships in heterogeneous data. J. Biomed. Inform. **52**, 438–447 (2014)
18. Le, T.D., Zhang, J., Liu, L., Liu, H., Li, J.: miRLAB: an R based dry lab for exploring miRNA-mRNA regulatory relationships. PLoS One **10**(12), e0145386 (2015)