INDEED R package for cancer biomarker candidate selection

Chaohui Xu and Yiming Zuo

2018-04-20

The biomarker candidates selected by **INDEED** lead to more accurate survival time prediction compared with those selected by differential expression (DE) analysis and differential network (DN) analysis.

Introduction

Differential expression (DE) analysis is commonly used to identify biomarker candidates that have significant changes in their expression levels between distinct biological groups. One drawback of DE analysis is that it only considers the changes on single biomolecular level. In differential network (DN) analysis, network is typically built based on the correlation and biomarker candidates are selected by investigating the network topology. However, correlation tends to generate over-complicated networks and the selection of biomarker candidates purely based on network topology ignores the changes on single biomolecule level. Thus, we have proposed a novel method INDEED, which considers both the changes on single biomolecular and network levels by integrating DE and DN analysis. INDEED has been published in Methods journal (PMID: 27592383). This is the R package that implements the algorithm.

This R package will generate a csv file containing information such as p-values, node degree and activity score for each biomolecule. A higher activity score indicates that the corresponding biomolecule has more neighbors connceted in the differential network and their p-values are more statistically significant. It will also generate a csv file for the differential network created by INDEED.

The **INDEED** package doesn't get loaded automatically, so remember to load it first:

```
library(INDEED)
```

`glasso` function will also be loaded as **INDEED** depends on it to obtain the sparse differential network.

Using the function `select_sig()`

To use the function `sig_select()`, users will need to have these data sets in hand:

- (**x**) A data frame that contains the expression level of individual biomolecule from two biologically disparate groups (p * n)
- (**class_label**) A binary array with group 1 labeled as 0 and group 2 as 1
- (**id**) An array that includes the corresponding ID for each biomolecule
- (**p_val**) The path to csv file containing p-values obtained from DE analysis (optional)

The demo data is presented in the **Demo Data** section below. It will be automatically loaded with the package.

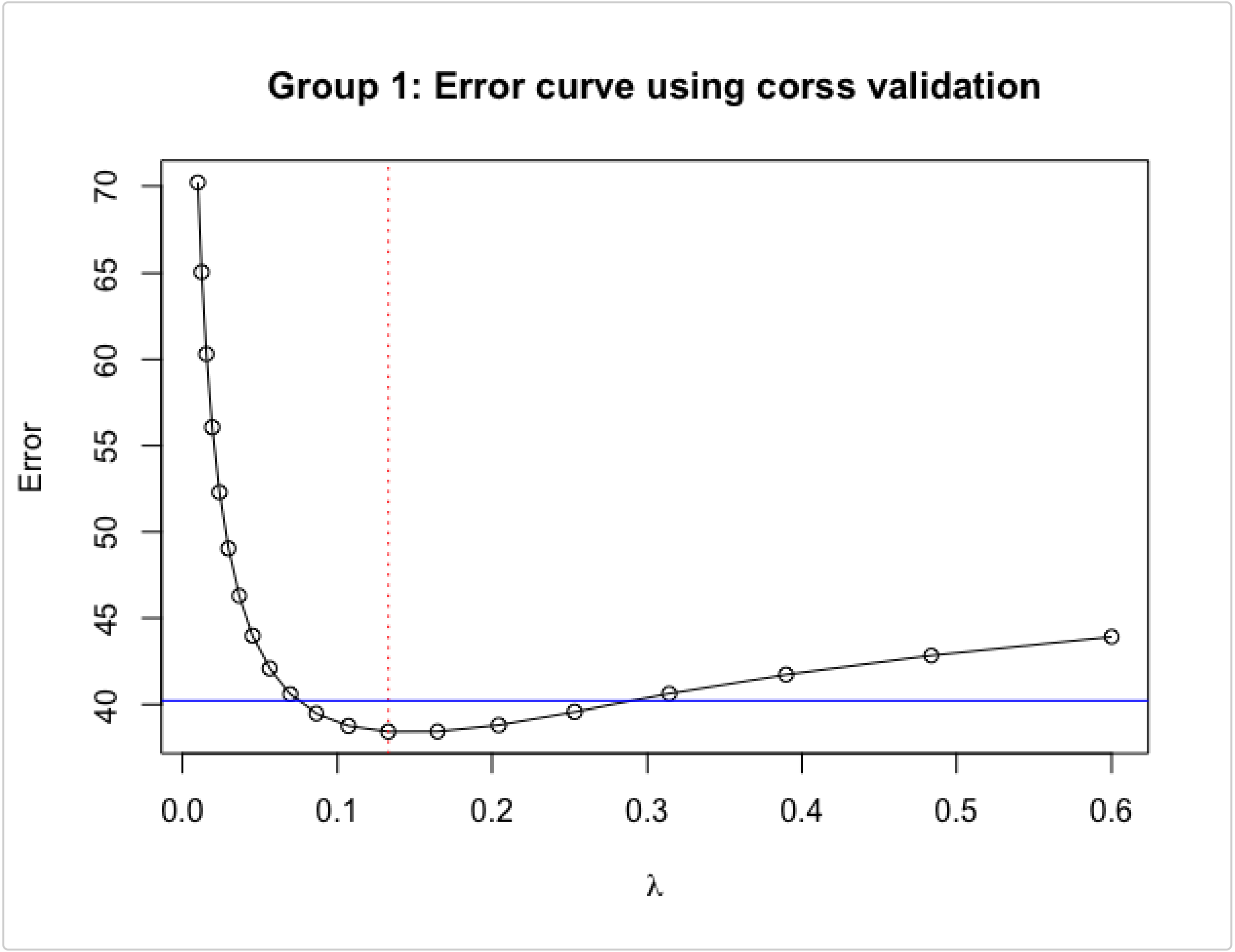
Now, users can test `select_sig()` to see how it works within sample data sets. One good thing about this package is that it gives users great flexibility. For example, although correlation tends to generate over-complicated networks, `select_sig()` still allows users to choose between using correlation or partial correlation to generate networks. Setting `partial = TRUE` will generate a sparse differential network using partial correlation. By setting `partial = FALSE`, users will get network based on correlation. If users forgot to specify the `partial` parameter, there will be an error message: `Error in if (partial == TRUE) { : argument is of length zero}`. In addtion, once `partial = TRUE` is specified, users will be able to interact with console to select their desired regularization parameter **rho** in order to obtain the sparse differential network. If answered [n] (Default), another question will pop up asking the users whether they want to select the **rho** based on the minimum rule [m] or one standard error rule [o] (Default). Also, network generated based on correlation can be either Pearson or Spearman correlation. The default is Pearson correlation. `method = "spearman"` will use Spearman instead. For both correlation and partial correlation, users get to determine the permutation times based on their preferences.

Remember, a file containing p-values is optional as p-values could be calculated using logistic regression if **p_val** is set to NULL or not specified.

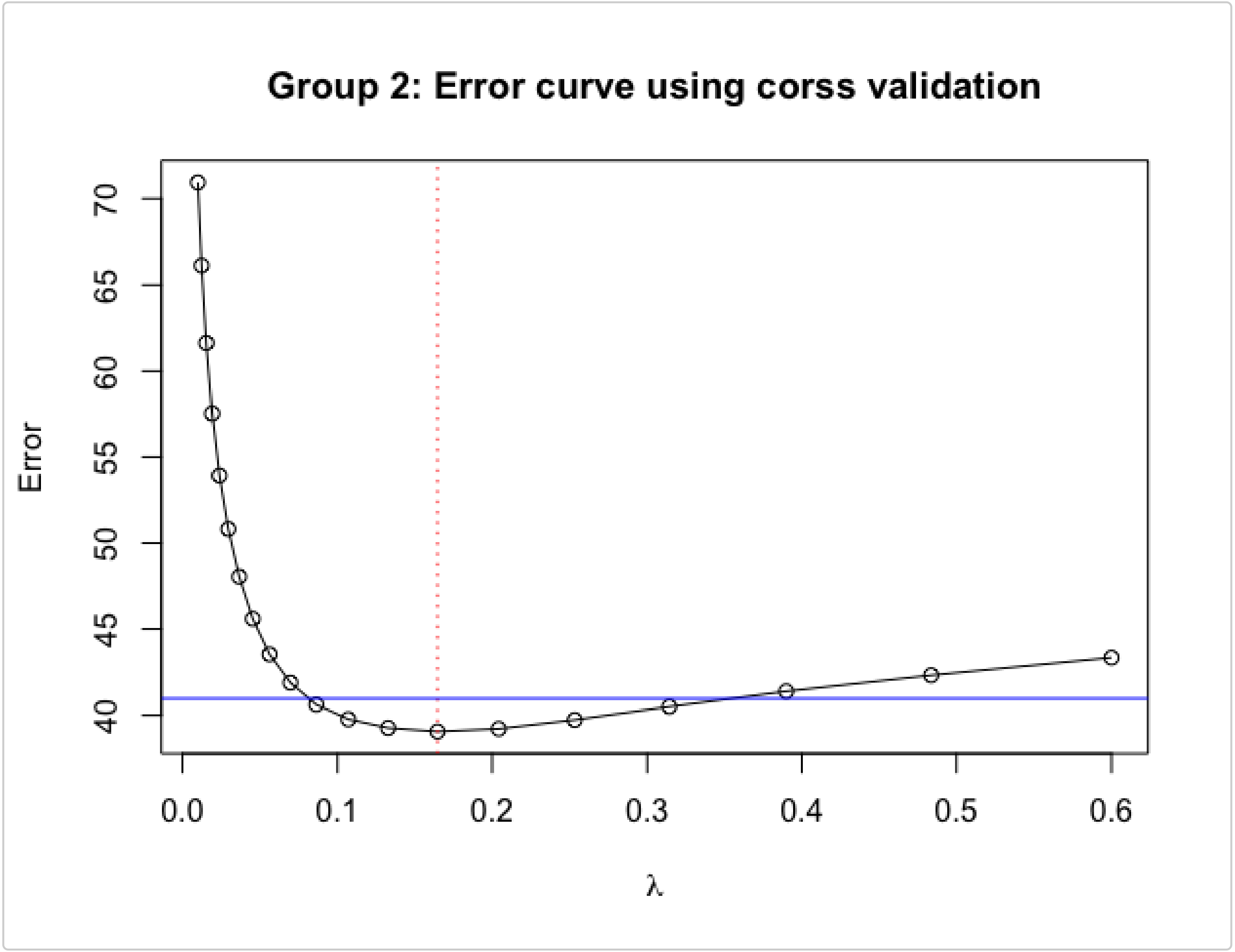
The following example demonstrates how to use `select_sig` function:

```
select_sig(x = Met_GU, class_label = Met_Group_GU, id = Met_name_GU, partial = TRUE)
```

In this case, the sparse differential network is based on partial correlation and p-value for each biomolecule is calculated for users. Also, **rho** is picked based on one standard error rule and number of permutations is set to 1000.



```
[1] "The list of rhos for group 1:"
 [1] 0.01000000 0.01240472 0.01538770 0.01908801 0.02367814 0.02937207 0.03643522 0.04519687
0.05606544 0.06954760 0.08627184 0.10701779
[13] 0.13275256 0.16467581 0.20427570 0.25339825 0.31433340 0.38992173 0.48368692 0.60000000
Choose your own regularization parameter rho for group 1 (Default: n)? [y/n]: n
rho based on minimum rule/ rho based on one standard error rule (Default: o) [m/o]: o
[1] 0.2533983
```



```
[1] "The list of rhos for group 2:"
 [1] 0.01000000 0.01240472 0.01538770 0.01908801 0.02367814 0.02937207 0.03643522 0.04519687
0.05606544 0.06954760 0.08627184 0.10701779
[13] 0.13275256 0.16467581 0.20427570 0.25339825 0.31433340 0.38992173 0.48368692 0.60000000
Choose your own regularization parameter rho for group 2 (Default: n)? [y/n]: n
rho based on minimum rule/ rho based on one standard error rule (Default: o) [m/o]: o
[1] 0.2533983
```

```
Enter your desired number of permutations to build differential network using partial
correlation [Default: 1000]: 1000
```

The table below is part of the output:

ID	P-value	Node Degree	Activity Score
C00009	0.273302332130741	1	1.64694048411187
C00022	0.384859184217185	8	7.72073652063631
C00025	0.511023518870014	8	8.57028806953701
C00049	0.872369870500983	3	3.9851046343879
C00064	0.0326255206776045	2	3.45077916647801
C00065	0.161752613981253	10	10.3210203575861
C00086	0.911023614391541	5	3.0277470561921
C00097	0.552005500173557	11	12.7253003101729
C00124	0.579530368526951	3	4.83165468807371
C00148	0.843295333958664	6	7.21740057265386

There are more examples in the **More Examples** section below.

Demo Data

- Met_GU (**x**): The data set contains 120 patients as columns and 39 metabolites as rows. Here, 29 rows and 110 columns are omitted.

Met_GU

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
-1.1778429	-0.6524507	0.1130101	0.3273883	-0.8159722	0.9169098	-0.1060636	-0.1486893	-0.7536426	1.9331369
-0.7446555	-0.8403552	1.2275791	1.4884276	0.9581165	0.2217579	0.5787392	-0.0405991	-0.3448051	-0.3943420
1.0200524	1.6526556	0.4660893	1.4657142	1.1549580	0.6665652	-0.0223597	-0.2524002	0.6314481	-0.2927764
0.4043534	0.4216086	0.3728297	0.4413724	0.4105573	0.3923992	0.3448359	0.6497466	0.3820917	0.3832617
1.2702685	1.5406950	-0.1213972	1.0226981	-1.4156816	0.0233863	2.1908966	-0.8078932	0.1743634	1.2832645
0.0485523	0.6102747	1.0018852	0.8012087	0.0337508	0.2927706	0.2096389	0.2585413	0.8692107	-0.5259235
0.6894522	0.3685145	0.7439457	0.5439377	-0.3837053	-1.9448212	0.9260460	0.6932116	0.5020424	1.5166784

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
1.2353851	0.6499381	1.1800297	0.8708025	0.0966225	-0.0139761	0.3602818	0.9736498	0.0021787	0.0364516
0.8014495	0.7095510	0.8441953	0.6402270	0.6989732	0.3359723	-0.1608252	0.2642710	0.1746253	0.8041863
-0.2097359	-0.3054575	1.8151690	0.0355987	-0.1561255	-0.3512114	-0.2574128	0.5083526	1.6624811	0.5962978

- pvalue_M_GU (i.e. **p_val** change in expression level of a single biomolecule between distinct biological group). Here, only the first 10 metabolites are shown.

P-values

KEGG.ID	p.value
C00009	0.4889976
C00022	0.8252445
C00025	0.9965415
C00049	0.4885672
C00064	0.0154086
C00065	0.7779342
C00086	0.7748844
C00097	0.5782100
C00124	0.3257471
C00148	0.9000034

- Met_Group_GU (i.e. **class_label**, group 1:0; group 2:1)

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0	0	0	0	0	0	0	0	0	0

An error message: `Error in Data[testIndexes, ] : subscript out of bounds` will occur if this **class_lable** is missing.

- Met_name_GU (i.e. **id**)

x
C00009
C00022
C00025
C00049
C00064
C00065
C00086
C00097
C00124
C00148

Also, if the users want the p-values to be calculated for them without providing this data, an error message will occur:
`Error in colnames(X_df)[1:ncol(X_df) - 1] <- Met_name : replacement has length zero`.

More Examples

Example 1

```
select_sig(x = Met_GU, class_label = Met_Group_GU, id = Met_name_GU, partial = FALSE)
```

By calling the above function, the differential network is obtained based on Pearson correlation coefficient. And the table below is part of the csv file output.

ID	P-value	Node Degree	Activity Score
C00009	0.273302332130741	2	1.7586879856516
C00022	0.384859184217185	0	0.868977963217195
C00025	0.511023518870014	2	3.27057926672721

C00049	0.872369870500983	1	1.54014499875783
C00064	0.0326255206776045	3	4.58094579084526
C00065	0.161752613981253	1	1.75108125516533
C00086	0.911023614391541	2	1.80739783498726
C00097	0.552005500173557	0	0.594757619634409
C00124	0.579530368526951	3	2.31622854753933
C00148	0.843295333958664	3	4.96404691696128

Example 2

```
select_sig(x = Met_GU, class_label = Met_Group_GU, Met_name = Met_name_GU, partial = FALSE,
method = "spearman", p_val = "data/pvalue_M_GU.csv")
```

By calling the above function, the differential network is obtained based on Spearman correlation coefficient. And the table below is part of the csv file output.

ID	P-value	Node Degree	Activity Score
C00009	0.488997612	1	1.09658805197093
C00022	0.825244477	0	0.220804734459591
C00025	0.996541506	3	5.08340875889463
C00049	0.488567238	4	4.66521168303724
C00064	0.015408576	5	7.04030425033912
C00065	0.777934199	1	0.898685171604638
C00086	0.774884414	3	2.65975664197908
C00097	0.578209969	0	0.556001295717367
C00124	0.325747138	5	4.69595104881907
C00148	0.900003382	3	3.38789107113181