Code Testing Documentation

1 fci

This function accepts as arguments the following:

- `NumericMatrix true_dag` - This matrix represents one of two things. If the user supplies the true DAG adjacency matrix, then this will ensure that we obtain the correct neighborhood for the target node(s). This is useful in a simulation setting. If the user does not supply the true DAG adjacency matrix, then this matrix will encode the estimated neighborhood relationships for the target node(s) and its (their) neighbors.
- `arma::mat df` - This will hold the sample data we will use to estimate the local structure of our target node(s).
- `NumericVector targets` - This vector will contain all of the target nodes we are considering. The target values correspond to columns in `df` and columns/rows in `true_dag`.
- `StringVector names` - A vector containing the names of the nodes. If not provided, default values will be given.
- `int lmax` - An integer providing the maximum size of a potential separating set checked by our algorithm. Default value is 3.
- `double signif_level` - This value provides the significance level for our statistical tests to determine conditional independence. Default value is 0.05.
- `bool verbose` - This determines whether or not we will produce all of the output for each step of the algorithm.

At the beginning of this function, we identify the number of targets and the number of total nodes in the network. We then call the `fci_setup` function.

2 fci_setup

This function can be found in `skeletonHelpersEfficient.cpp`. Its dependencies include `sharedFunctions.h` and `pCorTest.h`. Inputs include a matrix representing either the true DAG in the population version or a graph adjacency matrix that approximates all necessary neighborhood information, a vector of the target nodes under consideration, a vector of Strings containing the names of the target nodes, a constant integer containing the largest value of ℓ being considered to determine the size of the potential separating sets, and a boolean to determine whether or not the output is verbose.

3 get_multiple_neighbors_from_dags

Implemented in `sharedFunctions.cpp`. Tested in `test_setup.R`. This takes a vector of targets as an input and loops through each target, calling `get_neighbors_from_dag`

4 union_

This function takes in two vectors as arguments and returns a vector that represents the union of the two vectors taken as sets. Tested in `cpptests.cpp`

5 std::sort

Sorts a NumericVector in place. Tested in `cpptests.cpp`.

6 std::fill

Fills a NumericMatrix with a particular value specified as an argument of the function. Tested in `cpptests.cpp`.

7 fill_diag

Fills the diagonal elements of a matrix with a user-specified value. Tested in `cpptests.cpp`.

8 create_conditioning_sets_efficient

Tested in `test_efficientSset.R`.

9 get_skeleton_total

Can be found in `main.cpp`.

10 combn_cpp

This translates the `combn` function from R into a C++ function. It is tested in `test_skeletonhelpers.R`.

11 create

This is tested in cpptests.cpp

12 setdiff

This is tested in cpptests.cpp

13 get_current_edges

this is tested in test_skeletonhelpers.R

14 check_separation_sample_efficient

tested in test_skeletonhelpers.R

15 get_skeleton_target

Can be found in `main.cpp`.

16 std::map

All tests for these can be found in cpptests.cpp

16.1 find

16.2 insert

17 intersect

tested in cpptests.cpp

18 get_potential_sep

Written in skeletonHelpersEfficient.cpp. Tested to some extent in test_skeletonhelpers.R.

19 check_separation_sample_efficient_target

written in skeletonHelpersEfficient.cpp. Needs to be tested

Local FCI Tests

Tests can be found in `test_lfci.R`. We are using the BN "asia" to test the functionality of this class. The interface between the class functions and the testing script in R is found in `testLocalFCI.cpp`.

- `initializeLocalFCI`
  Here we initialize the class and print all of the elements belonging to the object. The output from the test is printed below. The first part is the output from the structure, and the output following that is from the `print_elements` function.

  Nodes in order according to the R dataframe:

Node	R Index	C++ Index
asia	1	0
tub	2	1
smoke	3	2
lung	4	3
bronc	5	4
either	6	5
xray	7	6
dysp	8	7

```
> initializeLocalFCI(asiaDAG,asiadf,3,node_names)
There is (are) 1 target(s).
Targets: lung
Target: 3
FUNCTION get_neighbors_from_dag. Node 3
Call from get_neighbors_from_dag. Node 2 is a parent.
Call from get_neighbors_from_dag. Node 5 is a child.
Call from get_neighbors_from_dag. We are evaluating the following child: 5
Call from get_neighbors_from_dag. Node 1 is a potential spouse of node 3.
Neighbors of node 3: 1, 2, 5

Total Neighborhood:
1, 2, 5
There are 8 nodes in the DAG.
There are 4 nodes in the neighborhoods we are considering.
All nodes being considered: 1 2 3 5
Our starting matrix is 4x4.
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
```

```
Our initial separating sets:
S[[1]][[1]] = nan S[[1]][[2]] = nan S[[1]][[3]] = nan S[[1]][[5]] = nan
S[[2]][[1]] = nan S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[5]] = nan
S[[3]][[1]] = nan S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[5]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = nan S[[5]][[3]] = nan S[[5]][[5]] = nan
Element mapping for efficient ordering:
1 0
2 1
3 2
5 3


p: 8
n: 500
N: 4
Number of Targets: 1
Node names: asia tub smoke lung bronc either xray dysp
lmax: 3
verbose: 1
Nodes under consideration: 1 2 3 5
Ctilde:
Our Ctilde matrix is 4x4
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
Our DAG matrix is
0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Separating Set Values:
S[[1]][[1]] = nan S[[1]][[2]] = nan S[[1]][[3]] = nan S[[1]][[5]] = nan
S[[2]][[1]] = nan S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[5]] = nan
S[[3]][[1]] = nan S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[5]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = nan S[[5]][[3]] = nan S[[5]][[5]] = nan
First and last elements of the dataset: -0.641447 0.198884
```

Next is the population version of the algorithm.

```
> initializeLocalFCIPop(asiaDAG,3,node_names)
There are 1 targets.
Targets: lung
Target: 3
FUNCTION get_neighbors_from_dag. Node 3
Call from get_neighbors_from_dag. Node 2 is a parent.
Call from get_neighbors_from_dag. Node 5 is a child.
Call from get_neighbors_from_dag. We are evaluating the following child: 5
Call from get_neighbors_from_dag. Node 1 is a potential spouse of node 3.
Neighbors of node 3: 1, 2, 5

Total Neighborhood:
1, 2, 5
There are 8 nodes in the DAG.
There are 4 nodes in the neighborhood.
All nodes being considered: 1 2 3 5
Our starting matrix is 4x4.
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0



Our initial separating sets:
S[[1]][[1]] = nan S[[1]][[2]] = nan S[[1]][[3]] = nan S[[1]][[5]] = nan
S[[2]][[1]] = nan S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[5]] = nan
S[[3]][[1]] = nan S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[5]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = nan S[[5]][[3]] = nan S[[5]][[5]] = nan
Element mapping for efficient ordering:
1 0
2 1
3 2
5 3



p: 8
n: 0
N: 4
Number of Targets: 1
Node names: asia tub smoke lung bronc either xray dysp
lmax: 3
verbose: 1
Nodes under consideration: 1 2 3 5
Ctilde:
```

```
Our Ctilde matrix is 4x4
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
Our DAG matrix is
0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Separating Set Values:
S[[1]][[1]] = nan S[[1]][[2]] = nan S[[1]][[3]] = nan S[[1]][[5]] = nan
S[[2]][[1]] = nan S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[5]] = nan
S[[3]][[1]] = nan S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[5]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = nan S[[5]][[3]] = nan S[[5]][[5]] = nan
```

- `checkSkeletonTotal` - ensures that the first stage of the PC algorithm works. We continue where we left off in the first bullet point. We first recall that we have changed the labeling in order to speed up calculations.

Node	True Number	Efficient Number
Tub	1	0
Smoke	2	1
Lung	3	2
Either	5	3

Table 1: Numbering is 0-indexed to accord with C++ convention

```
> result_amat <- checkSkeletonTotal(asiaDAG,asiadf,3,node_names)
There is (are) 1 target(s).
Targets: lung
Target: 3
FUNCTION get_neighbors_from_dag. Node 3
Call from get_neighbors_from_dag. Node 2 is a parent.
Call from get_neighbors_from_dag. Node 5 is a child.
Call from get_neighbors_from_dag. We are evaluating the following child: 5
Call from get_neighbors_from_dag. Node 1 is a potential spouse of node 3.
Neighbors of node 3: 1, 2, 5

Total Neighborhood:
1, 2, 5
```

```
There are 8 nodes in the DAG.
There are 4 nodes in the neighborhoods we are considering.
All nodes being considered: 1 2 3 5
Our starting matrix is 4x4.
0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0



Our initial separating sets:
S[[1]][[1]] = nan S[[1]][[2]] = nan S[[1]][[3]] = nan S[[1]][[5]] = nan
S[[2]][[1]] = nan S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[5]] = nan
S[[3]][[1]] = nan S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[5]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = nan S[[5]][[3]] = nan S[[5]][[5]] = nan
Element mapping for efficient ordering:
1 0
2 1
3 2
5 3



The value of l is 0
The value of i is 0 (tub)
The value of j is 1 (smoke)
The p-value is 0.0664439
tub is separated from smoke (p-value>0.01)
The value of j is 2 (lung)
The p-value is 0.631592
tub is separated from lung (p-value>0.01)
The value of j is 3 (either)
The p-value is 2.96476e-07
The value of i is 1 (smoke)
The value of j is 2 (lung)
The p-value is 3.57115e-12
The value of j is 3 (either)
The p-value is 3.62121e-05
The value of i is 2 (lung)
The value of j is 3 (either)
The p-value is 2.97187e-109
The value of i is 3 (either)
The value of l is 1
The value of i is 0 (tub)
The value of j is 3 (either)
```

```
There are 2 neighbors.
Efficient Setup: 0 -> 1 | 3 -> 5 | k (True Vals): 3 (lung)
The p-value is 9.35856e-14
tub is NOT separated from either by node(s): lung  (p-value<0.01)
Efficient Setup: 0 -> 1 | 3 -> 5 | k (True Vals): 2 (smoke)
The p-value is 2.23369e-08
tub is NOT separated from either by node(s): smoke  (p-value<0.01)
The value of i is 1 (smoke)
The value of j is 2 (lung)
There is 1 neighbor.
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 5 (either)
The p-value is 6.33935e-09
smoke is NOT separated from lung by node(s): either  (p-value<0.01)
The value of j is 3 (either)
There are 2 neighbors.
Efficient Setup: 1 -> 2 | 3 -> 5 | k (True Vals): 3 (lung)
The p-value is 0.0944292
smoke is separated from either by node(s): lung  (p-value>0.01)
The value of i is 2 (lung)
The value of j is 3 (either)
There are 2 neighbors.
Efficient Setup: 2 -> 3 | 3 -> 5 | k (True Vals): 2 (smoke)
The p-value is 5.05153e-105
lung is NOT separated from either by node(s): smoke  (p-value<0.01)
Efficient Setup: 2 -> 3 | 3 -> 5 | k (True Vals): 1 (tub)
The p-value is 3.10595e-117
lung is NOT separated from either by node(s): tub  (p-value<0.01)
The value of i is 3 (either)
The value of l is 2
The value of i is 0 (tub)
The value of j is 3 (either)
The value of i is 1 (smoke)
The value of j is 2 (lung)
The value of i is 2 (lung)
The value of j is 3 (either)
There are 2 neighbors.
Efficient Setup: 2 -> 3 | 3 -> 5 | k (True Vals): 2 1 (smoke tub)
The p-value is 4.9879e-111
lung is NOT separated from either by node(s): smoke tub  (p-value<0.01)
The value of i is 3 (either)
The value of l is 3
The value of i is 0 (tub)
The value of j is 3 (either)
The value of i is 1 (smoke)
The value of j is 2 (lung)
```

```
The value of i is 2 (lung)
The value of j is 3 (either)
The value of i is 3 (either)


Values after Total Skeleton Run

p: 8
n: 500
N: 4
Number of Targets: 1
Node names: asia tub smoke lung bronc either xray dysp
lmax: 3
verbose: 1
Nodes under consideration: 1 2 3 5
Ctilde:
Our Ctilde matrix is 4x4
0 0 0 1
0 0 1 0
0 1 0 1
1 0 1 0
Our DAG matrix is
0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Separating Set Values:
S[[1]][[1]] = nan S[[1]][[2]] = -1 S[[1]][[3]] = -1 S[[1]][[5]] = nan
S[[2]][[1]] = -1 S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[5]] = 3
S[[3]][[1]] = -1 S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[5]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = 3 S[[5]][[3]] = nan S[[5]][[5]] = nan
First and last elements of the dataset: -0.641447 0.198884
```

Checking the first two tests to ensure accuracy:

```
> cor.test(~tub+smoke,data = asiadf)

Pearson's product-moment correlation

data:  tub and smoke
t = -1.8393, df = 498, p-value = 0.06646
```

```
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.168620185  0.005586615
sample estimates:
        cor
-0.08214421

> cor.test(~tub+lung,data = asiadf)

Pearson's product-moment correlation

data:  tub and lung
t = 0.48001, df = 498, p-value = 0.6314
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.06631098  0.10898971
sample estimates:
       cor
0.02150465

> cor.test(~tub+either,data = asiadf)

Pearson's product-moment correlation

data:  tub and either
t = -5.1762, df = 498, p-value = 3.292e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.3075474 -0.1410547
sample estimates:
       cor
-0.2259504
```

Now for the partial correlations:

```
> ppcor::pcor.test(asiadf$tub,asiadf$either,asiadf$lung)
    estimate      p.value statistic   n gp  Method
1 -0.3225578 1.517173e-13 -7.597003 500  1 pearson

> ppcor::pcor.test(asiadf$smoke,asiadf$either,asiadf$lung)
    estimate    p.value statistic   n gp  Method
1 0.07495583 0.09441775  1.675742 500  1 pearson
```

Two Targets Test

```
> result_amat <- checkSkeletonTotal(asiaDAG,asiadf,c(3,4),node_names)
```

Node	True Number	Efficient Number
Tub	1	0
Smoke	2	1
Lung	3	2
Bronc	4	3
Either	5	4
Dysp	7	5

Table 2: Numbering is 0-indexed to accord with C++ convention

```
There is (are) 2 target(s).
Targets: lung bronc
Target: 3
FUNCTION get_neighbors_from_dag. Node 3
Call from get_neighbors_from_dag. Node 2 is a parent.
Call from get_neighbors_from_dag. Node 5 is a child.
Call from get_neighbors_from_dag. We are evaluating the following child: 5
Call from get_neighbors_from_dag. Node 1 is a potential spouse of node 3.
Neighbors of node 3: 1, 2, 5

Target: 4
FUNCTION get_neighbors_from_dag. Node 4
Call from get_neighbors_from_dag. Node 2 is a parent.
Call from get_neighbors_from_dag. Node 7 is a child.
Call from get_neighbors_from_dag. We are evaluating the following child: 7
Call from get_neighbors_from_dag. Node 5 is a potential spouse of node 4.
Neighbors of node 4: 2, 5, 7

Total Neighborhood:
1, 2, 5, 7
There are 8 nodes in the DAG.
There are 6 nodes in the neighborhoods we are considering.
All nodes being considered: 1 2 3 4 5 7
Our starting matrix is 6x6.
0 1 1 1 1 1
1 0 1 1 1 1
1 1 0 1 1 1
1 1 1 0 1 1
1 1 1 1 0 1
1 1 1 1 1 0



Our initial separating sets:
```

```
S[[1]][[1]] = nan S[[1]][[2]] = nan S[[1]][[3]] = nan S[[1]][[4]] = nan
S[[1]][[5]] = nan S[[1]][[7]] = nan
S[[2]][[1]] = nan S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[4]] = nan
S[[2]][[5]] = nan S[[2]][[7]] = nan
S[[3]][[1]] = nan S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[4]] = nan
S[[3]][[5]] = nan S[[3]][[7]] = nan
S[[4]][[1]] = nan S[[4]][[2]] = nan S[[4]][[3]] = nan S[[4]][[4]] = nan
S[[4]][[5]] = nan S[[4]][[7]] = nan
S[[5]][[1]] = nan S[[5]][[2]] = nan S[[5]][[3]] = nan S[[5]][[4]] = nan
S[[5]][[5]] = nan S[[5]][[7]] = nan
S[[7]][[1]] = nan S[[7]][[2]] = nan S[[7]][[3]] = nan S[[7]][[4]] = nan
S[[7]][[5]] = nan S[[7]][[7]] = nan
Element mapping for efficient ordering:
1 0
2 1
3 2
4 3
5 4
7 5


The value of l is 0
The value of i is 0 (tub)
The value of j is 1 (smoke)
The p-value is 0.0664439
tub is separated from smoke (p-value>0.01)
The value of j is 2 (lung)
The p-value is 0.631592
tub is separated from lung (p-value>0.01)
The value of j is 3 (bronc)
The p-value is 0.579122
tub is separated from bronc (p-value>0.01)
The value of j is 4 (either)
The p-value is 2.96476e-07
The value of j is 5 (dysp)
The p-value is 0.0049716
The value of i is 1 (smoke)
The value of j is 2 (lung)
The p-value is 3.57115e-12
The value of j is 3 (bronc)
The p-value is 6.44217e-19
The value of j is 4 (either)
The p-value is 3.62121e-05
The value of j is 5 (dysp)
The p-value is 8.722e-05
```

```
The value of i is 2 (lung)
The value of j is 3 (bronc)
The p-value is 0.00672051
The value of j is 4 (either)
The p-value is 2.97187e-109
The value of j is 5 (dysp)
The p-value is 9.83333e-14
The value of i is 3 (bronc)
The value of j is 4 (either)
The p-value is 0.0431059
bronc is separated from either (p-value>0.01)
The value of j is 5 (dysp)
The p-value is 6.74668e-84
The value of i is 4 (either)
The value of j is 5 (dysp)
The p-value is 9.48354e-32
The value of i is 5 (dysp)
The value of l is 1
The value of i is 0 (tub)
The value of j is 4 (either)
There are 3 neighbors.
Efficient Setup: 0 -> 1 | 4 -> 5 | k (True Vals): 3 (lung)
The p-value is 9.35856e-14
tub is NOT separated from either by node(s): lung  (p-value<0.01)
Efficient Setup: 0 -> 1 | 4 -> 5 | k (True Vals): 7 (dysp)
The p-value is 1.75872e-05
tub is NOT separated from either by node(s): dysp  (p-value<0.01)
Efficient Setup: 0 -> 1 | 4 -> 5 | k (True Vals): 2 (smoke)
The p-value is 2.23369e-08
tub is NOT separated from either by node(s): smoke  (p-value<0.01)
The value of j is 5 (dysp)
There are 4 neighbors.
Efficient Setup: 0 -> 1 | 5 -> 7 | k (True Vals): 5 (either)
The p-value is 0.669315
tub is separated from dysp by node(s): either  (p-value>0.01)
The value of i is 1 (smoke)
The value of j is 2 (lung)
There are 3 neighbors.
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 5 (either)
The p-value is 6.33935e-09
smoke is NOT separated from lung by node(s): either  (p-value<0.01)
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 4 (bronc)
The p-value is 1.73265e-10
smoke is NOT separated from lung by node(s): bronc  (p-value<0.01)
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 7 (dysp)
```

```
The p-value is 1.85744e-19
smoke is NOT separated from lung by node(s): dysp  (p-value<0.01)
The value of j is 3 (bronc)
There are 3 neighbors.
Efficient Setup: 1 -> 2 | 3 -> 4 | k (True Vals): 5 (either)
The p-value is 5.30428e-18
smoke is NOT separated from bronc by node(s): either  (p-value<0.01)
Efficient Setup: 1 -> 2 | 3 -> 4 | k (True Vals): 3 (lung)
The p-value is 3.36192e-17
smoke is NOT separated from bronc by node(s): lung  (p-value<0.01)
Efficient Setup: 1 -> 2 | 3 -> 4 | k (True Vals): 7 (dysp)
The p-value is 1.40787e-17
smoke is NOT separated from bronc by node(s): dysp  (p-value<0.01)
The value of j is 4 (either)
There are 4 neighbors.
Efficient Setup: 1 -> 2 | 4 -> 5 | k (True Vals): 7 (dysp)
The p-value is 9.84372e-13
smoke is NOT separated from either by node(s): dysp  (p-value<0.01)
Efficient Setup: 1 -> 2 | 4 -> 5 | k (True Vals): 3 (lung)
The p-value is 0.0944292
smoke is separated from either by node(s): lung  (p-value>0.01)
The value of j is 5 (dysp)
There are 3 neighbors.
Efficient Setup: 1 -> 2 | 5 -> 7 | k (True Vals): 5 (either)
The p-value is 2.3399e-12
smoke is NOT separated from dysp by node(s): either  (p-value<0.01)
Efficient Setup: 1 -> 2 | 5 -> 7 | k (True Vals): 4 (bronc)
The p-value is 0.0018733
smoke is NOT separated from dysp by node(s): bronc  (p-value<0.01)
Efficient Setup: 1 -> 2 | 5 -> 7 | k (True Vals): 3 (lung)
The p-value is 4.7219e-12
smoke is NOT separated from dysp by node(s): lung  (p-value<0.01)
The value of i is 2 (lung)
The value of j is 3 (bronc)
There are 3 neighbors.
Efficient Setup: 2 -> 3 | 3 -> 4 | k (True Vals): 7 (dysp)
The p-value is 9.74475e-37
lung is NOT separated from bronc by node(s): dysp  (p-value<0.01)
Efficient Setup: 2 -> 3 | 3 -> 4 | k (True Vals): 2 (smoke)
The p-value is 0.868413
lung is separated from bronc by node(s): smoke  (p-value>0.01)
The value of j is 4 (either)
There are 4 neighbors.
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 4 (bronc)
The p-value is 7.07433e-108
```

```
lung is NOT separated from either by node(s): bronc  (p-value<0.01)
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 2 (smoke)
The p-value is 5.05153e-105
lung is NOT separated from either by node(s): smoke  (p-value<0.01)
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 7 (dysp)
The p-value is 1.21991e-94
lung is NOT separated from either by node(s): dysp  (p-value<0.01)
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 1 (tub)
The p-value is 3.10595e-117
lung is NOT separated from either by node(s): tub  (p-value<0.01)
The value of j is 5 (dysp)
There are 3 neighbors.
Efficient Setup: 2 -> 3 | 5 -> 7 | k (True Vals): 2 (smoke)
The p-value is 4.8985e-21
lung is NOT separated from dysp by node(s): smoke  (p-value<0.01)
Efficient Setup: 2 -> 3 | 5 -> 7 | k (True Vals): 5 (either)
The p-value is 0.0810771
lung is separated from dysp by node(s): either  (p-value>0.01)
The value of i is 3 (bronc)
The value of j is 5 (dysp)
There are 2 neighbors.
Efficient Setup: 3 -> 4 | 5 -> 7 | k (True Vals): 2 (smoke)
The p-value is 3.28744e-82
bronc is NOT separated from dysp by node(s): smoke  (p-value<0.01)
Efficient Setup: 3 -> 4 | 5 -> 7 | k (True Vals): 5 (either)
The p-value is 1.86754e-176
bronc is NOT separated from dysp by node(s): either  (p-value<0.01)
The value of i is 4 (either)
The value of j is 5 (dysp)
There are 4 neighbors.
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 3 (lung)
The p-value is 1.21054e-19
either is NOT separated from dysp by node(s): lung  (p-value<0.01)
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 1 (tub)
The p-value is 7.29264e-30
either is NOT separated from dysp by node(s): tub  (p-value<0.01)
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 2 (smoke)
The p-value is 1.15967e-39
either is NOT separated from dysp by node(s): smoke  (p-value<0.01)
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 4 (bronc)
The p-value is 8.04073e-114
either is NOT separated from dysp by node(s): bronc  (p-value<0.01)
The value of i is 5 (dysp)
The value of l is 2
The value of i is 0 (tub)
```

```
The value of j is 4 (either)
There are 2 neighbors.
Efficient Setup: 0 -> 1 | 4 -> 5 | k (True Vals): 3 7 (lung dysp)
The p-value is 6.06083e-12
tub is NOT separated from either by node(s): lung dysp  (p-value<0.01)
The value of i is 1 (smoke)
The value of j is 2 (lung)
There are 3 neighbors.
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 7 4 (dysp bronc)
The p-value is 2.32256e-08
smoke is NOT separated from lung by node(s): dysp bronc  (p-value<0.01)
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 7 5 (dysp either)
The p-value is 3.23618e-08
smoke is NOT separated from lung by node(s): dysp either  (p-value<0.01)
Efficient Setup: 1 -> 2 | 2 -> 3 | k (True Vals): 4 5 (bronc either)
The p-value is 3.38235e-08
smoke is NOT separated from lung by node(s): bronc either  (p-value<0.01)
The value of j is 3 (bronc)
There are 2 neighbors.
Efficient Setup: 1 -> 2 | 3 -> 4 | k (True Vals): 3 7 (lung dysp)
The p-value is 1.66923e-06
smoke is NOT separated from bronc by node(s): lung dysp  (p-value<0.01)
The value of j is 5 (dysp)
There are 3 neighbors.
Efficient Setup: 1 -> 2 | 5 -> 7 | k (True Vals): 3 4 (lung bronc)
The p-value is 0.53713
smoke is separated from dysp by node(s): lung bronc  (p-value>0.01)
The value of i is 2 (lung)
The value of j is 4 (either)
There are 3 neighbors.
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 1 2 (tub smoke)
The p-value is 4.9879e-111
lung is NOT separated from either by node(s): tub smoke  (p-value<0.01)
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 1 7 (tub dysp)
The p-value is 2.39758e-102
lung is NOT separated from either by node(s): tub dysp  (p-value<0.01)
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 2 7 (smoke dysp)
The p-value is 3.88973e-81
lung is NOT separated from either by node(s): smoke dysp  (p-value<0.01)
The value of i is 3 (bronc)
The value of j is 5 (dysp)
There are 2 neighbors.
Efficient Setup: 3 -> 4 | 5 -> 7 | k (True Vals): 2 5 (smoke either)
The p-value is 9.14632e-161
bronc is NOT separated from dysp by node(s): smoke either  (p-value<0.01)
```

```
The value of i is 4 (either)
The value of j is 5 (dysp)
There are 3 neighbors.
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 1 3 (tub lung)
The p-value is 8.41815e-18
either is NOT separated from dysp by node(s): tub lung  (p-value<0.01)
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 1 4 (tub bronc)
The p-value is 1.59159e-110
either is NOT separated from dysp by node(s): tub bronc  (p-value<0.01)
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 3 4 (lung bronc)
The p-value is 1.28527e-59
either is NOT separated from dysp by node(s): lung bronc  (p-value<0.01)
The value of i is 5 (dysp)
The value of l is 3
The value of i is 0 (tub)
The value of j is 4 (either)
The value of i is 1 (smoke)
The value of j is 2 (lung)
The value of j is 3 (bronc)
The value of i is 2 (lung)
The value of j is 4 (either)
There are 3 neighbors.
Efficient Setup: 2 -> 3 | 4 -> 5 | k (True Vals): 1 2 7 (tub smoke dysp)
The p-value is 3.4474e-87
lung is NOT separated from either by node(s): tub smoke dysp  (p-value<0.01)
The value of i is 3 (bronc)
The value of j is 5 (dysp)
The value of i is 4 (either)
The value of j is 5 (dysp)
There are 3 neighbors.
Efficient Setup: 4 -> 5 | 5 -> 7 | k (True Vals): 1 3 4 (tub lung bronc)
The p-value is 3.62282e-56
either is NOT separated from dysp by node(s): tub lung bronc  (p-value<0.01)
The value of i is 5 (dysp)


Values after Total Skeleton Run

p: 8
n: 500
N: 6
Number of Targets: 2
Node names: asia tub smoke lung bronc either xray dysp
lmax: 3
verbose: 1
```

```
Nodes under consideration: 1 2 3 4 5 7
Ctilde:
Our Ctilde matrix is 6x6
0 0 0 0 1 0
0 0 1 1 0 0
0 1 0 0 1 0
0 1 0 0 0 1
1 0 1 0 0 1
0 0 0 1 1 0
Our DAG matrix is
0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Separating Set Values:
S[[1]][[1]] = nan S[[1]][[2]] = -1 S[[1]][[3]] = -1 S[[1]][[4]] = -1 S[[1]][[5]] = nan S
S[[2]][[1]] = -1 S[[2]][[2]] = nan S[[2]][[3]] = nan S[[2]][[4]] = nan S[[2]][[5]] = 3 S
S[[3]][[1]] = -1 S[[3]][[2]] = nan S[[3]][[3]] = nan S[[3]][[4]] = 2 S[[3]][[5]] = nan S
S[[4]][[1]] = -1 S[[4]][[2]] = nan S[[4]][[3]] = 2 S[[4]][[4]] = nan S[[4]][[5]] = -1 S[
S[[5]][[1]] = nan S[[5]][[2]] = 3 S[[5]][[3]] = nan S[[5]][[4]] = -1 S[[5]][[5]] = nan S
S[[7]][[1]] = 5 S[[7]][[2]] = 3 4 S[[7]][[3]] = 5 S[[7]][[4]] = nan S[[7]][[5]] = nan S[
First and last elements of the dataset: -0.641447 0.198884
```

Checking partial correlation calculation while conditioning on two variables

```
> ppcor::pcor.test(asiadf$smoke,asiadf$dysp,asiadf[,c("lung","bronc")])
    estimate   p.value statistic   n gp  Method
1 0.02773211 0.5369505 0.6178611 500  2 pearson
```

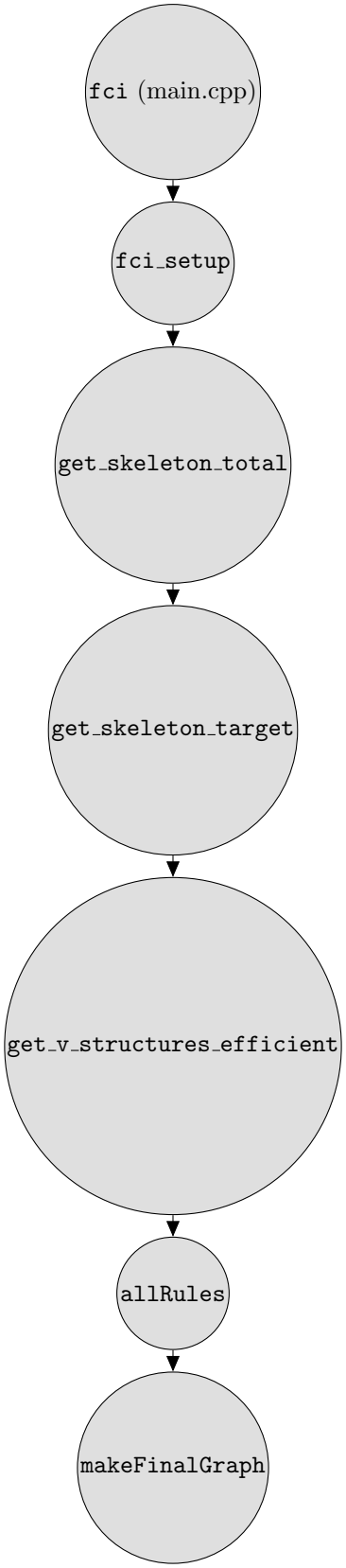
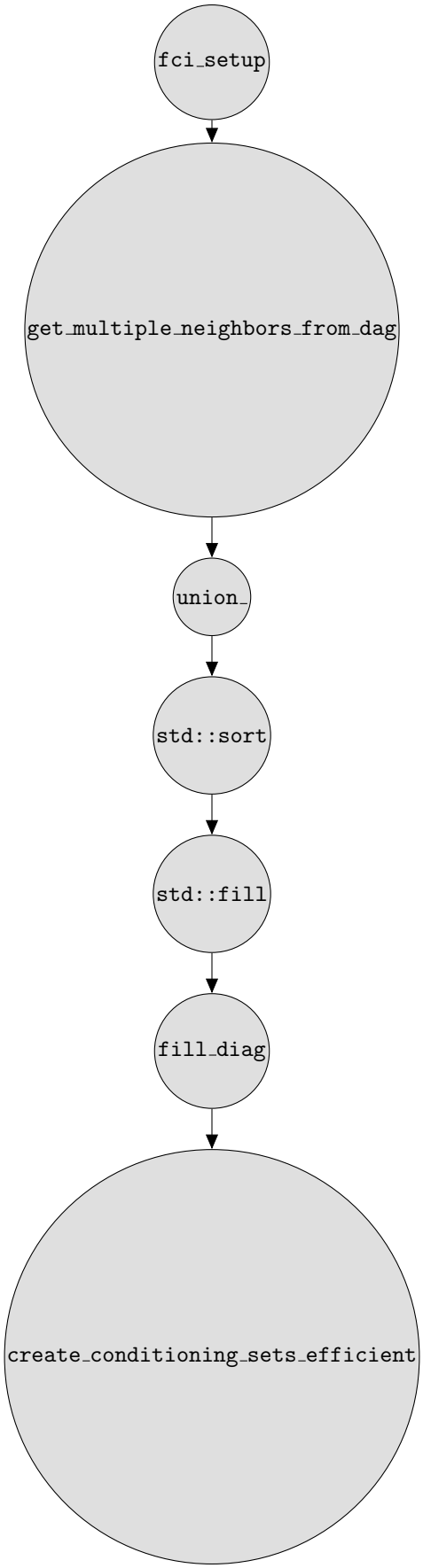
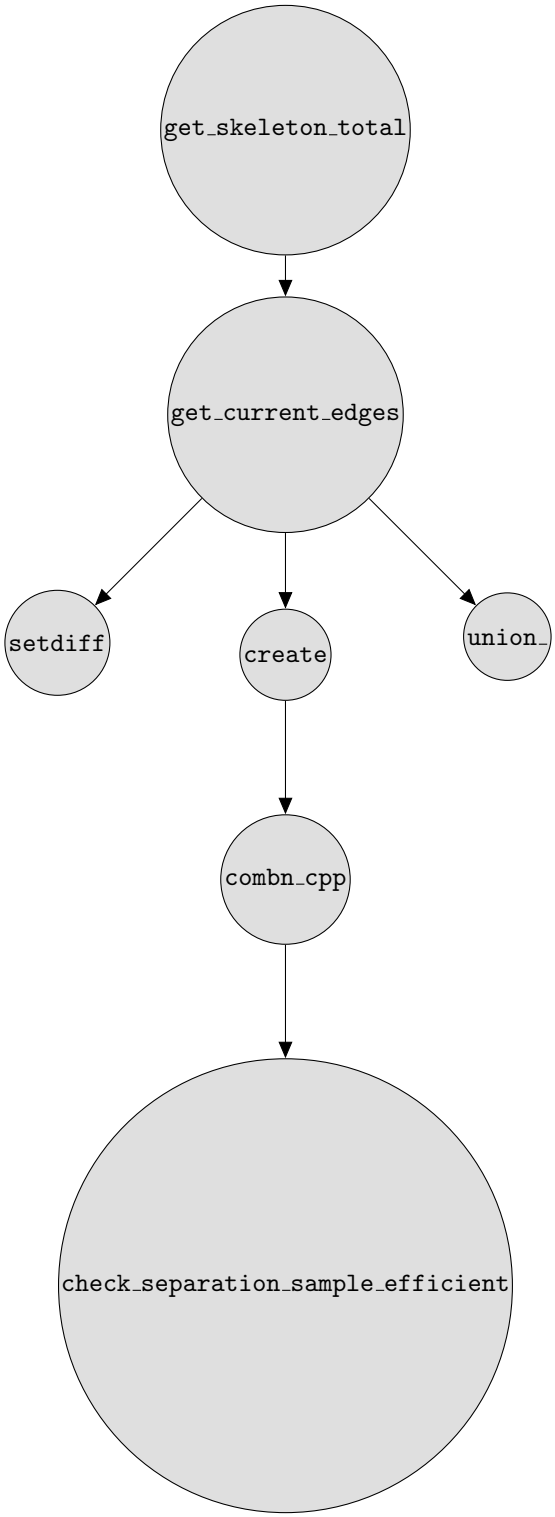
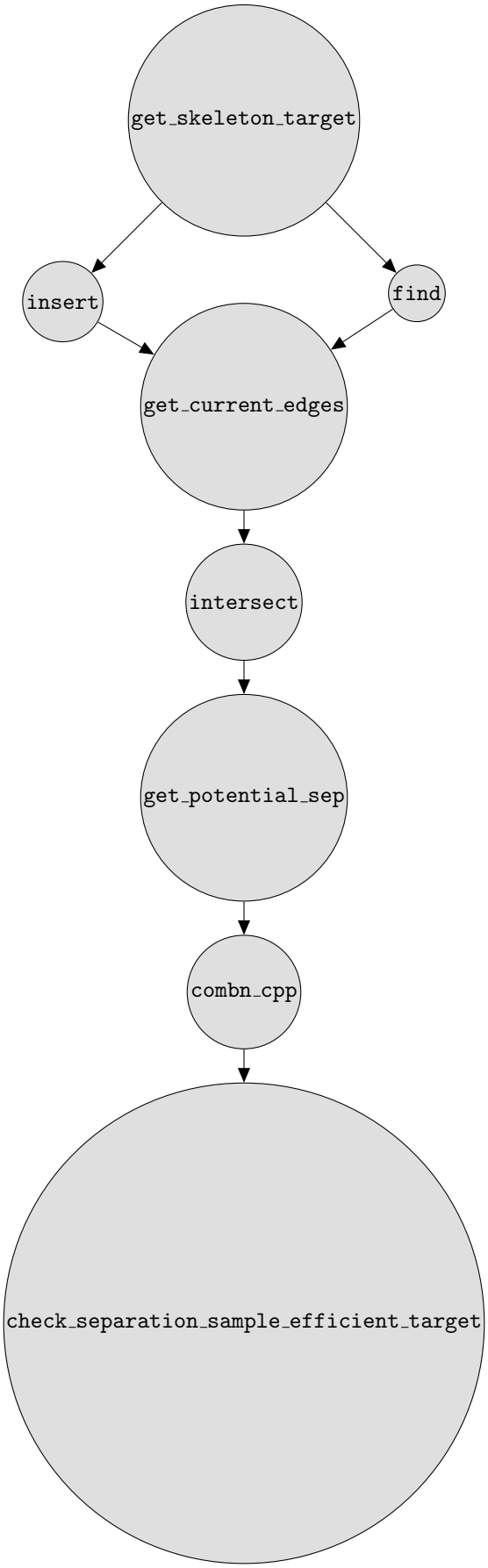


Figure 1: Flow of primary function `fic`


fci_setup
get_multiple_neighbors_from_dag
union_
std::sort
std::fill
fill_diag
create_conditioning_sets_efficient


get_skeleton_total
get_current_edges
setdiff
create
union_
combn_cpp
check_separation_sample_efficient


get_skeleton_target
insert
find
get_current_edges
intersect
get_potential_sep
combn_cpp
check_separation_sample_efficient_target

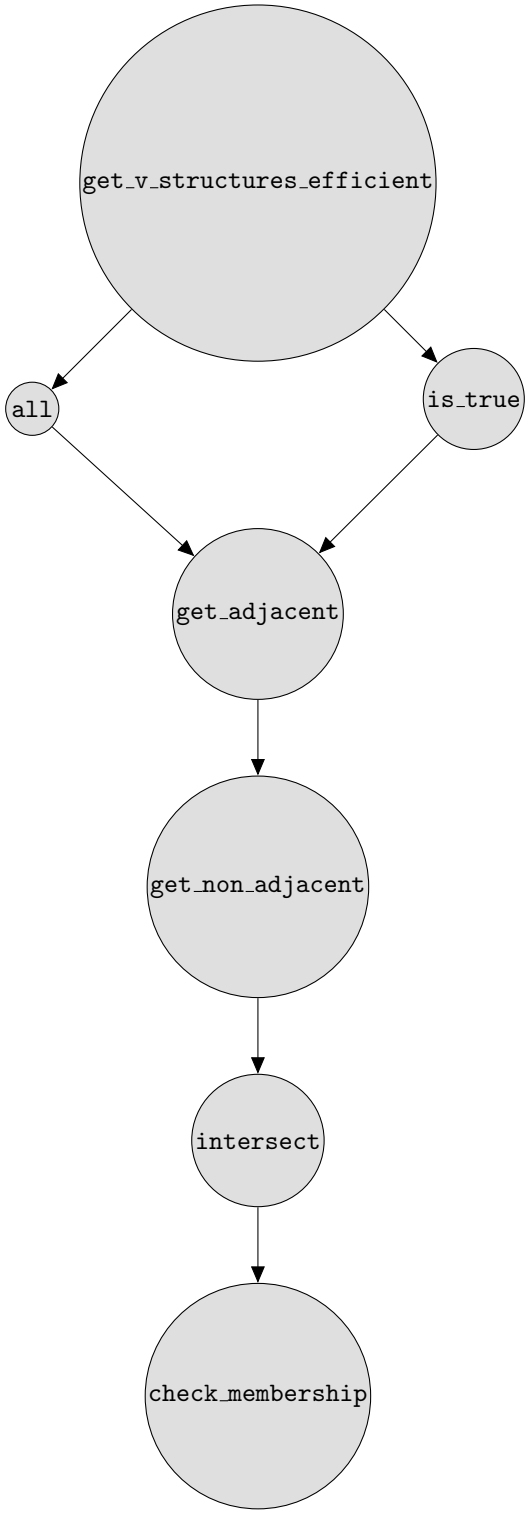


Figure 2: This function may be found in `vStructHelpers.cpp`