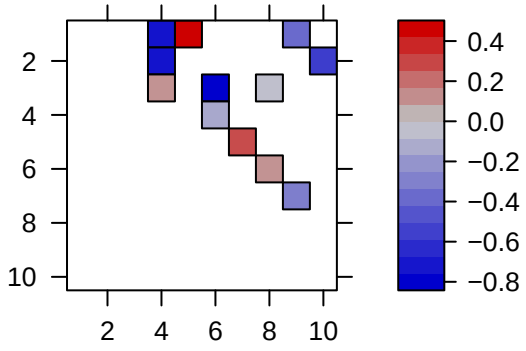
lattice_tests

Loading or creating the data

```
source('load_lattice_data.R')
```

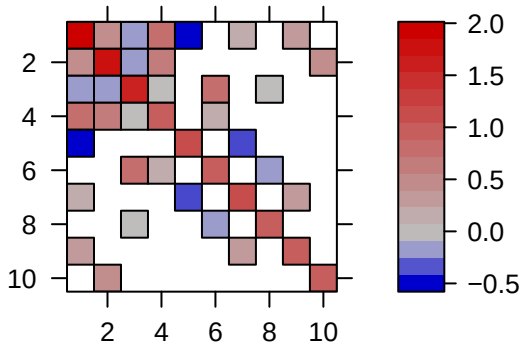
Plot B

```
image(Bet, sub="", xlab="", ylab="")
```



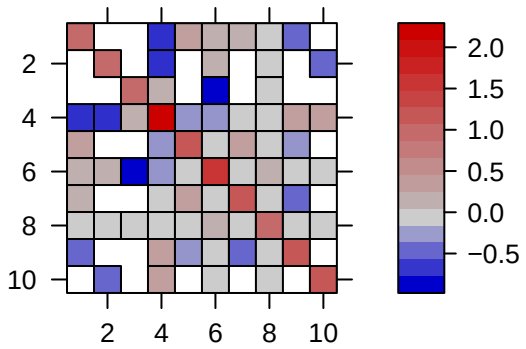
Plot of the inverse covariance matrix

```
image(Gamma, sub="", xlab="", ylab="")
```



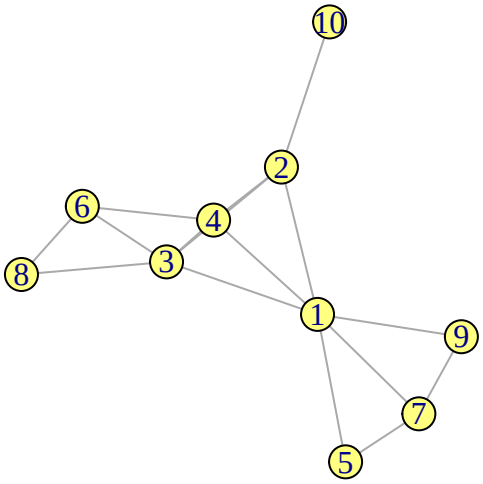
and the covariance matrix

```
image(Sig, sub="", xlab="", ylab="")
```



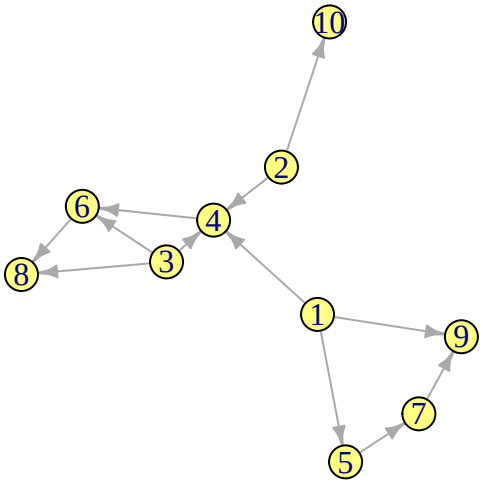
and the PCG

```
plot(pcg)
```



and the original DAG

```
plot(dag)
```



SEM coefficients and single lattice computations

An example of getting SEM coefficients $\beta_j(S)$

```
j <- 2
S <- c(3,4,7,8,10)
print(getParentCoefs2(j, S, Sig, threshold=T)$coefs,  col.names=T)
```

```
## 1 x 10 sparse Matrix of class "dgCMatrix"
##      1 2          3          4 5 6           7 8 9         10
## [1,] . . 0.04614923 -0.2884327 . . -0.02594821 . . -0.3218463
```

Computing an example lattice:

```
j <- 2
S <- all_nodes[-j]
lat <- computeLattice(j, c(3,4,10), Sig, all_nodes, should_sort=F)
lat
```

```
## {3,4,10} -- {3,4,10,6,8}  (4 sets)
```

The two sets shown are the minimum (m) and the maximum (M) of the lattice.

Checking whether a set belongs to a lattice:

```
isInLattice(c(6,3,8,4,10),lat)
```

```
## [1] TRUE
```

Another example (j is 2)

```
print(computeLattice(j, c(1,8,9,5,7), Sig, all_nodes), width=30)
```

```
##                           {1,8} --                    {1,8,9,5,7}  (8 sets)
```

Computing all subordinate lattices

Compute all the lattices subordinate to the lattice generated by `all_nodes[-j]` (i.e., have smaller minimums)

```
lattices <- computeAllSubordinateLattices(j, Sig, all_nodes, print.width = 25, VERB=2)
```

```
##                {1,3,4,10} --      {1,3,4,5,6,7,8,9,10} (32 sets)   S = {1,3,4,5,6,7,8,9,10}
##                   {1,3,4} --         {1,3,4,5,6,7,8,9} (32 sets)   S = {1,3,4}
##                      {10} --            {1,3,10,5,7,9} (32 sets)   S = {1,3,10}
##                  {1,4,10} --            {1,4,10,5,7,9}  (8 sets)   S = {1,4,10}
##                  {3,4,10} --              {3,4,10,6,8}  (4 sets)   S = {3,4,10}
##                        {} --              {1,3,5,7,9} (32 sets)   S = {1,3}
##                     {1,4} --              {1,4,5,7,9}  (8 sets)   S = {1,4}
##                     {3,4} --                {3,4,6,8}  (4 sets)   S = {3,4}
##                    {4,10} --                   {4,10}  (1 sets)   S = {4,10}
##                       {4} --                      {4}  (1 sets)   S = {4}
##
## Total number of sets covered = 154
```

and checking whether a set is in any of these:

```
is_set_in_lattices(c(1,3,7,8,9),lattices)
```

```
## [1] FALSE
```

Since this set is outside let us use it as root to generate more lattices:

```
lattices <- computeAllSubordinateLattices(j, Sig, all_nodes, root_S = c(1,3,7,8,9), print.width = 25, V
```

```
##                   {1,3,8} --            {1,3,7,8,9,5}  (8 sets)   S = {1,3,7,8,9}
##                       {} --              {1,3,5,7,9} (32 sets)   S = {1,3}
##                    {1,8} --              {1,8,5,7,9}  (8 sets)   S = {1,8}
##                    {3,8} --                    {3,8}  (1 sets)   S = {3,8}
##                      {8} --                      {8}  (1 sets)   S = {8}
##
## Total number of sets covered = 50
```

Let us change $j = 1$ and see what happens:

```
j <- 1
lattices <- computeAllSubordinateLattices(j, Sig, all_nodes, print.width = 30, VERB=2)
```

```
##                  {2,3,4,5,7,9} --          {2,3,4,5,6,7,8,9,10}  (8 sets)   S = {2,3,4,5,6,7,8,9,10}
##                      {2,3,4,5} --            {2,3,4,5,7,6,8,10} (16 sets)   S = {2,3,4,5,7}
##                    {2,3,4,5,9} --            {2,3,4,5,9,6,8,10}  (8 sets)   S = {2,3,4,5,9}
##                    {2,3,4,7,9} --            {2,3,4,7,9,6,8,10}  (8 sets)   S = {2,3,4,7,9}
##                      {5,7,9} --                {2,3,5,7,9,10}  (8 sets)   S = {2,3,5,7,9}
##                    {2,4,5,7,9} --                {2,4,5,7,9,10}  (2 sets)   S = {2,4,5,7,9}
##                    {3,4,5,7,9} --               {3,4,5,7,9,6,8}  (4 sets)   S = {3,4,5,7,9}
##                      {2,3,4} --                 {2,3,4,6,8,10}  (8 sets)   S = {2,3,4}
##                         {5} --                   {2,3,5,7,10} (16 sets)   S = {2,3,5}
##                      {2,4,5} --                   {2,4,5,7,10}  (4 sets)   S = {2,4,5}
##                      {3,4,5} --                  {3,4,5,6,7,8}  (8 sets)   S = {3,4,5}
##                    {2,3,4,9} --              {2,3,4,9,6,8,10}  (8 sets)   S = {2,3,4,9}
##                       {5,9} --                  {2,3,5,9,10}  (8 sets)   S = {2,3,5,9}
##                    {2,4,5,9} --                  {2,4,5,9,10}  (2 sets)   S = {2,4,5,9}
##                    {3,4,5,9} --                 {3,4,5,9,6,8}  (4 sets)   S = {3,4,5,9}
##                    {2,3,4,7} --              {2,3,4,7,6,8,10}  (8 sets)   S = {2,3,4,7}
```

```
##                         {7,9} --                 {2,3,7,9,10}  (8 sets)   S = {2,3,7,9}
##                       {2,4,7,9} --                 {2,4,7,9,10}  (2 sets)   S = {2,4,7,9}
##                       {3,4,7,9} --                {3,4,7,9,6,8}  (4 sets)   S = {3,4,7,9}
##                       {4,5,7,9} --                    {4,5,7,9}  (1 sets)   S = {4,5,7,9}
##                             {} --                     {2,3,10}  (8 sets)   S = {2,3}
##                          {2,4} --                     {2,4,10}  (2 sets)   S = {2,4}
##                          {3,4} --                    {3,4,6,8}  (4 sets)   S = {3,4}
##                          {4,5} --                      {4,5,7}  (2 sets)   S = {4,5}
##                            {9} --                   {2,3,9,10}  (8 sets)   S = {2,3,9}
##                        {2,4,9} --                   {2,4,9,10}  (2 sets)   S = {2,4,9}
##                        {3,4,9} --                  {3,4,9,6,8}  (4 sets)   S = {3,4,9}
##                        {4,5,9} --                      {4,5,9}  (1 sets)   S = {4,5,9}
##                            {7} --                   {2,3,7,10}  (8 sets)   S = {2,3,7}
##                        {2,4,7} --                   {2,4,7,10}  (2 sets)   S = {2,4,7}
##                        {3,4,7} --                  {3,4,7,6,8}  (4 sets)   S = {3,4,7}
##                        {4,7,9} --                      {4,7,9}  (1 sets)   S = {4,7,9}
##                            {4} --                         {4}  (1 sets)   S = {4}
##                        {4,9} --                       {4,9}  (1 sets)   S = {4,9}
##                        {4,7} --                       {4,7}  (1 sets)   S = {4,7}
##
## Total number of sets covered = 184
```

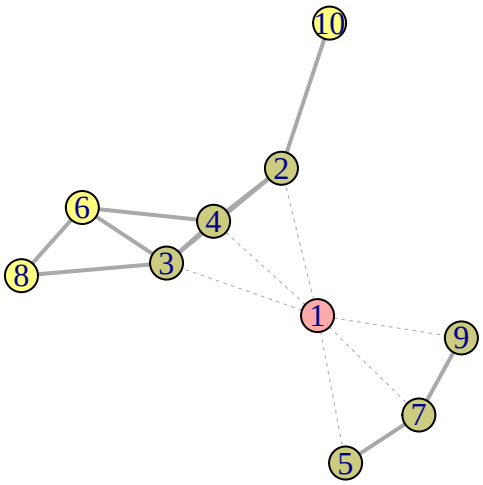
Verify component decomposition

```
#Randomly generate $j$ and $S \subset [d] \setminus j$
#S <- sample(1:d, sample(d-1,1))
#j <- sample(setdiff(all_nodes, S),1)
S <- c(2,3,4,5,7,9)
j <- 1
```

and verify the PCG connected component decomposition theorem (Theorem~?)

```
out <- verifyComponentDecomp(j, S, pcg, Sig, all_nodes, print.width = 25, should_sort=T)
```

```
##  Original:            {2,3,4,5,7,9} --      {2,3,4,5,6,7,8,9,10}  (8 sets)   S = {2,3,4,5,7,9}
##       S_1:                  {2,3,4} --            {2,3,4,6,8,10}  (8 sets)   S = {2,3,4}
##       S_2:                  {5,7,9} --            {2,3,5,7,9,10}  (8 sets)   S = {5,7,9}
##  verified:                      yes --                       yes
```



j is shown in red, and S in darker colors. The 2 component(s) after removing `j` are visible.

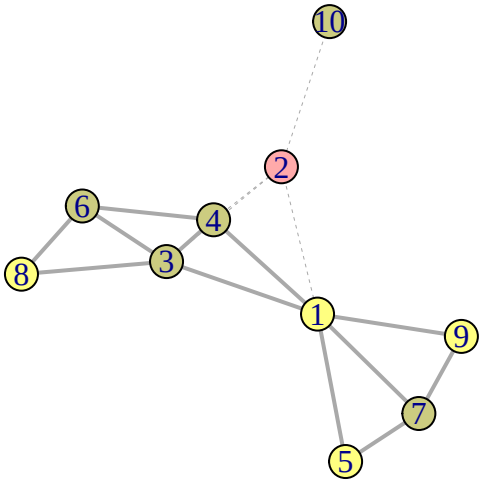
```
out$verified
```

```
## [1] TRUE
```

An example where this verification failed

```
j <- 2
S <- c(3,4,6,7,10)
out <- verifyComponentDecomp(j, S, pcg, Sig, all_nodes, print.width = 25, should_sort=T)
```

```
##  Original:              {3,4,7,10} --           {3,4,6,7,8,10}  (4 sets)   S = {3,4,6,7,10}
##       S_1:                 {3,4,7} --              {3,4,6,7,8}  (4 sets)   S = {3,4,6,7}
##       S_2:                    {10} --           {1,3,5,7,9,10} (32 sets)   S = {10}
##  verified:                     yes --                       no
```



What went wrong seems to be a failure of perfectness:

```
A <- c(10)
j <- 2
R <-  c(1,3,5,7,9)
verifyL2MarkovPerfectness(j, A, R, pcg, Sig, VERB=1)
```

```
## [1] "Graph failed, projection passed"
```

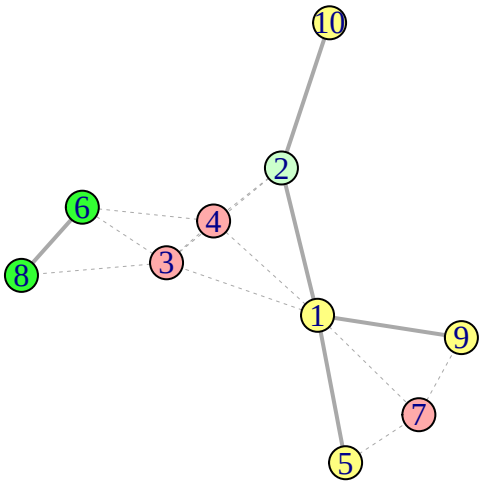
```
## [1] FALSE
```

```
# Another example
# S <- c(3,4,7)
# S <- c(6,5,9)
# S <- c(8,9)
# j <- 1
# out <- verifyComponentDecomp(j, S, pcg, Sig, all_nodes, print.width = 25, should_sort=T)
```

Graph seperation calculations

Find all the nodes seperated from A by S (here \$j = \$ 2)

```
S <- c(3,4,7)
A <- j
out <- all_seperated_from_A_by_S(A, S, pcg, plot.subg=T)
```



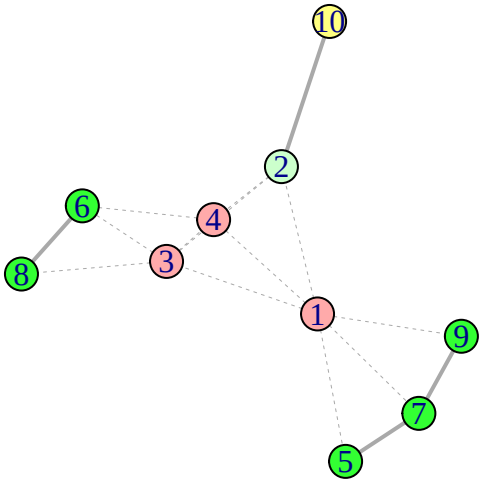

```
print(out$sep_set)
```

```
## {6,8}
```

The distance matrix above is between A and $[d] \setminus S$ in the graph obtained after removing S.

Another example

```
S <-  c(1,3,4)
A <- j
out <- all_seperated_from_A_by_S(A, S, pcg, plot.subg=T)
```




```
out
```

```
## $sep_set
## {5,6,7,8,9}
## $dist
##     5   6   7   8   9 10
```

```
## 2 Inf Inf Inf Inf Inf  1
```

Perfectness failure

```
A <- c(3,10)
S <- c(2,7)
B <- 1
```

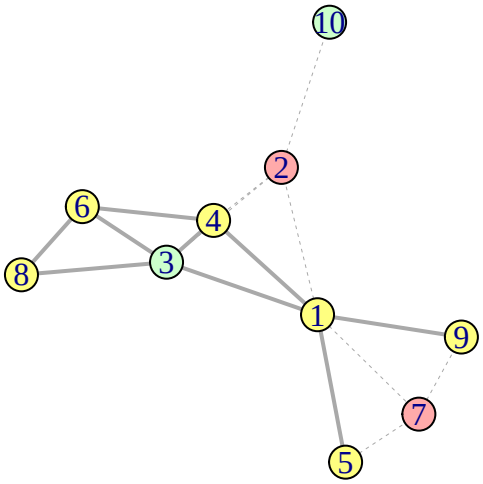
This verifies if $P_A P_S^{\perp} P_B = 0$:

```
A_and_B_proj_seperated_by_S(A, S, B, Sig, threshold=T)
```

```
## $verified
## [1] TRUE
##
## $the_matrix
## 1 x 2 sparse Matrix of class "dgCMatrix"
##
## [1,] . .
```

This verfies $A - S - B$ in the graph:

```
A_and_B_graph_seperated_by_S(A, S, B, pcg, plot.subg=T)
```




```
## [1] FALSE
```

This verifies Markov perfectness for triplet (A, S, B) (basically whether the answer to the two questions above match):

```
verifyL2MarkovPerfectness(A, S, B, pcg, Sig, VERB=1)
```

```
## [1] "Graph failed, projection passed"
```

```
## [1] FALSE
```