Is it a DAG?

2023-09-23

Contents

Comparing different algorithms to find a permutation that, applied to both rows and columns of a square matrix with zeros on the diagonal, results in a lower triangular matrix.

1 *John's function*

```r
isLowerTriangular <- function(X){
  all(0 == X[upper.tri(X)])
}

triangularize <- function(X){
  if (is.null(rownames(X))) rownames(X) <- 1:nrow(X)
  if (isLowerTriangular(X)) return(rownames(X))
  tri <- function(X){ # recursion helper function
    if (length(X) == 0) return()
    diag(X) <- 0
    roots <- which(rowSums(X) == 0)
    c(rownames(X)[roots], tri(X[-roots, -roots, drop=FALSE]))
  }
  perm <- tri(X)
  if (length(perm) != nrow(X)) stop("matrix cannot be triangularized")
  perm
}
```

2 *Georges' function*

```r
to_dag <- function(mat){
  # check that mat has unique column and row names
  find_leaves <- function(m) {   # wasteful because we only need for find one
                                 # 'orphan', not all possible 'orphans'
    sumabs <- function(x) sum(abs(x))
    diag(m) <- 0 # in case they're used for epsilons
    which(apply(m,2,sumabs)==0)
  }
  if(nrow(mat) != ncol(mat)) stop('matrix must be square')
  if(any(sort(colnames(mat)) != sort(rownames(mat)))) stop('colnames not same as rownames')
  if(length(unique(colnames(mat))) != length(colnames(mat))) stop('names not unique')
  mat <- mat[colnames(mat), colnames(mat)]
  if(sum(abs(mat[row(mat) < col(mat)])) == 0) {
    # matrix already lower diagonal
    class(mat) <- unique(c('dag', class(mat)))
    return(mat)
  }
  ret <- mat
  dag_perm <- rep('', nrow(mat))
  for(i in 1:nrow(ret)) {
    ll <- find_leaves(mat)
    if(length(ll) == 0) return(FALSE)
    dag_perm[i] <- names(ll[1])
    if(i < nrow(ret)) mat <- mat[-ll[1],-ll[1], drop = FALSE]
  }
  ret <- ret[rev(dag_perm),rev(dag_perm)]
  class(ret) <- unique(c('dag', class(ret)))
  ret
}
```

3 Georges’ function #2

This function exploits in-place replacement of elements of a matrix and borrows form John’s practice of avoiding passing arguments to a function defined within a function

```
to_dag <- function(mat){
  find_leaves <- function(m) {   # wasteful because we only need for find one
    # 'orphan', not all possible 'orphans'
    sumabs <- function(x) sum(abs(x))
    diag(m) <- 0 # in case they're used for epsilons
    which(apply(m,2,sumabs)==0)
  }
  if(nrow(mat) != ncol(mat)) stop('matrix must be square')
  if(any(sort(colnames(mat)) != sort(rownames(mat)))) stop('colnames not same as rownames')
  if(length(unique(colnames(mat))) != length(colnames(mat))) stop('names not unique')
  mat <- mat[colnames(mat), colnames(mat)]
  if(sum(abs(mat[row(mat) < col(mat)])) == 0) {
    # matrix already lower diagonal
    class(mat) <- unique(c('dag', class(mat)))
    return(mat)
  }
  ret <- mat
  diag(mat) <- 0
  dag_perm <- rep('', nrow(mat))
  for(i in 1:nrow(ret)) {

## CONTINUE %>% HEERE TO %>% IMPROVE %>% FUNCTION %>% SO %>% IT ------
## MAKES REFERENCES IN PLACE ------

    ll <- find_leaves(mat)
    if(length(ll) == 0) return(FALSE)
    dag_perm[i] <- names(ll[1])
    if(i < nrow(ret)) mat <- mat[-ll[1],-ll[1], drop = FALSE]
  }
  ret <- ret[rev(dag_perm),rev(dag_perm)]
  class(ret) <- unique(c('dag', class(ret)))
  ret
}
```

4 Random triangularizable matrix

```
rdag <- function(size) {
  x <- matrix(0,size,size)
  x[lower.tri(x)] <- rnorm(size*(size-1)/2)
  perm <- sample(size)
  x[perm, perm]
}
```

5 Timing tests

```
testn <- function(fun, size, reps = 1000) {
  system.time(
    lapply(seq_len(reps), function(i) fun(rdag(size)))
  )
}
```

```
testn(triangularize, 3)
```

```
      user  system elapsed
     0.063   0.000   0.063
```

```
testn(to_dag, 3)
```

```
      user  system elapsed
```

```
     0.059   0.000   0.059
```

```
testn(triangularize, 5)
```

```
      user  system elapsed
     0.061   0.000   0.061
```

```
testn(to_dag, 5)
```

```
      user  system elapsed
     0.045   0.000   0.045
```

```
testn(triangularize, 20)
```

```
      user  system elapsed
     0.213   0.000   0.213
```

```
testn(to_dag, 20)
```

```
      user  system elapsed
     0.052   0.000   0.052
```

```
testn(triangularize, 100)
```

```
      user  system elapsed
     4.795   0.004   4.799
```

```
testn(to_dag, 100)
```

```
      user  system elapsed
     0.229   0.000   0.229
```