Package 'unifDAG'

May 8, 2026

Type Package

Title Uniform Sampling of Directed Acyclic Graphs

Version 1.0.4

Date 2024-02-06

Maintainer Markus Kalisch <kalisch@stat.math.ethz.ch>

Author Markus Kalisch [aut, cre],
Manuel Schuerch [ctb]

Description Uniform sampling of Directed Acyclic Graphs (DAG) using exact enumeration by relating each DAG to a sequence of outpoints (nodes with no incoming edges) and then to a composition of integers as suggested by Kuipers, J. and Moffa, G. (2015) <doi:10.1007/s11222-013-9428-y>.

License GPL (>= 2)

Encoding UTF-8

Imports graph, gmp, stats, methods

Suggests Rgraphviz, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-06 16:00:02 UTC

Contents

unifDAG *Uniform Sampling of Directed Acyclic Graphs (DAG)*

Description

Uniform sampling of a labelled directed acyclic graph (DAG) with combinatorial enumeration.

Usage

```
unifDAG (n, weighted=FALSE, wFUN=list(runif, min=0.1, max=1))
unifDAG.approx(n, n.exact=20, weighted=FALSE, wFUN=list(runif,min=0.1,max=1))
```

Arguments

n	integer larger than 1, indicating the number of nodes in the DAG. unifDAG can only be used for n up to 100. For larger n, use unifDAG.approx.
weighted	logical indicating if weights of the edges are computed according to wFUN.
wFUN	a function for computing the weights of the edges in the DAG. It takes as first argument a number of edges m for which it returns a vector of length m containing the weights. Alternatively, it could be a list consisting of the function in the first entry and of further arguments of the function in the additional entries. The default (only if weighted is true)) is a uniform weight between 0.1 and 1. See the examples.
n.exact	an integer, at least n and between 2 and 100, denoting the number of nodes up to which the exact method is used, followed by an approximation for larger numbers up to n. See details on the quality of the approximation.

Details

A (weighted) random graph with n nodes is uniformly drawn over the space of all labelled DAGs with n nodes. The main idea of these two methods is to first sample a random sequence of outpoints, that is, nodes without incoming edges. This sequence is then used to construct an adjacency matrix, which is converted to the final DAG. The presented methods differ only in the approach to find this sequence of outpoints.

The method unifDAG builds the random sequence of outpoints based on precomputed enumeration tables.

The method unifDAG.approx executes unifDAG up to the number n.exact, for larger number of nodes an approximation is used instead. The default of n.exact = 20 (40) should get the approximation within the uniformity limits of a 32 (64) bit integer sampler. See reference for more details.

Value

A graph object of class graphNEL.

Note

The main advantage of these algorithms is that they operate on the space of DAGs instead of the space of undirected graphs with an additional phase of orienting the edges. With this approach the unintentional bias towards sparse graphs, for instance occurring when sampling adjacency matrices, can be eliminated.

Author(s)

Markus Kalisch (<kalisch@stat.math.ethz.ch>) and Manuel Schuerch.

References

Jack Kuipers and Giusi Moffa (2015) Uniform random generation of large acyclic digraphs. *Statistics and Computing* **25(2)**, 227–242, Springer; doi:10.1007/s112220139428y

Examples

```
set.seed(123)
dag1 <- unifDAG(n=10)
dag2 <- unifDAG.approx(n=10, n.exact=5)

dag <- unifDAG(n=5)
if (require("Rgraphviz")) plot(dag)
dag@edgeData   ## note the constant weights

dag <- unifDAG(n=5,weighted=TRUE)
if (require("Rgraphviz")) plot(dag)
dag@edgeData   ## note the uniform weights between 0.1 and 1

wFUN <- function(m,lB,uB) { runif(m,lB,uB) }
dag <- unifDAG(n=5,weighted=TRUE,wFUN=list(wFUN,1,4))
dag@edgeData   ## note the uniform weights between 1 and 4
```

Index