

Package: GWnorm (via r-universe)

October 5, 2026

Type Package

Title G-Wishart Normalising Constants for Gaussian Graphical Models

Version 1.0.2

Date 2026-07-07

Author Ching Wong [aut], Jack Kuipers [aut, cre]

Maintainer Jack Kuipers <jack.kuipers@bsse.ethz.ch>

Description Computes G-Wishart normalising constants through a Fourier approach. Either exact analytical results, numerical integration or Monte Carlo estimation are employed. Details at C. Wong, G. Moffa and J. Kuipers (2024), <[doi:10.48550/arXiv.2404.06803](https://doi.org/10.48550/arXiv.2404.06803)>. Also includes approximations of the ratio of normalising constants, see details at C. Wong, G. Moffa and J. Kuipers (2025), <[doi:10.48550/arXiv.2503.13046](https://doi.org/10.48550/arXiv.2503.13046)>.

License GPL (>= 2)

Depends R (>= 4.0.0)

Imports igraph, BDgraph, CholWishart, MASS, mvtnorm, hypergeo, gsl, Rcpp (>= 1.1.1)

LinkingTo Rcpp, RcppEigen

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Config/pak/sysreqs libglpk-dev libgsl0-dev libxml2-dev pari-gp

Repository <https://jackkuipers.r-universe.dev>

Date/Publication 2026-07-07 16:30:02 UTC

RemoteUrl <https://github.com/cran/GWnorm>

RemoteRef HEAD

RemoteSha b33a330ac6aa2b9242b19fdc7f9c42fedad14356

Contents

C_GtoI_G	2
check_prime_connected	3
I_G_BD	3
I_G_chordal	4
I_G_complete	5
I_G_MC	6
I_G_ratio_approx	7
I_G_ratio_approx_prime	8
I_G_special	9
I_Gnorm	11
I_GtoC_G	12
is_6_cycle	13
is_k_partite	13
Iss_cmat	14
Iss_mat	15
PD_complete	15
prime_decomp	16
Index	17

C_GtoI_G

G-Wishart normalising constant reparametrisation the other way

Description

This function returns the log of the G-Wishart normalising constant $I_G(\beta, D) = \int [S^p_{++}(G)] \det(K)^\beta \exp(-\text{tr}(KD)) dK$ from the transformed version $\log(C_G(\delta, D)) = \int [S^p_{++}(G)] \det(K)^{(\delta-2)/2} \exp(-\text{tr}(KD)/2) dK$

Usage

```
C_GtoI_G(graph, CGval, delta)
```

Arguments

graph	An igraph object, corresponding to a prime connected graph
CGval	A number $\log(C_G(\delta, D))$
delta	A constant > 0

Value

A number $\log(I_G(\beta, D))$

check_prime_connected *Prime and connected*

Description

This function just checks if a graph is connected and prime

Usage

```
check_prime_connected(graph, check_prime = TRUE)
```

Arguments

graph	An igraph object
check_prime	Boolean flag to check primality too (default TRUE)

Value

A boolean, TRUE if connected and prime

I_G_BD *G Wishart normalising constant with BDgraph*

Description

This function is a wrapper for BDgraph to compare and to avoid the NOTE in the package checks since we only had BDgraph in the examples

Usage

```
I_G_BD(graph, beta, D, n_samp = 1000, C_G_flag = FALSE)
```

Arguments

graph	An igraph object, corresponding to a prime connected graph
beta	A constant > 0
D	A p by p real symmetric positive definite matrix
n_samp	number of sample points used in MC integration
C_G_flag	boolean whether to return C_G instead of I_G

Value

A number $\log(I_G(\text{beta}, D))$ [or $\log(C_G(\text{delta}, D))$]

I_G_chordal

*G Wishart normalising constant for chordal graphs***Description**

This function returns the log of transformed G-Wishart normalising constant $I_G(\beta, D) = \int [S^p]_{++}(G) \det(K)^\beta \exp(-\text{tr}(KD)) dK$ for chordal graphs G , using explicit formula (cliques / separators).

Usage

```
I_G_chordal(
  graph,
  beta,
  D = NULL,
  cliques = NULL,
  separators = NULL,
  ratio = FALSE
)
```

Arguments

graph	An igraph object, corresponding to a chordal graph
beta	A constant > 0
D	A p by p real symmetric positive definite matrix (default, NULL for identity matrix)
cliques	A list of cliques (if known)
separators	A list of separators (if known)
ratio	Whether to output just the ratio compared to the value with I (default FALSE)

Value

A number $\log(I_G(\beta, D))$

Examples

```
set.seed(42)
p <- 5
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
beta <- 1
adj <- matrix(0, nrow = p, ncol = p)
adj[1,c(2:5)] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # chordal completion of 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_chordal(graph, beta, D)
# compare with BDgraph gnorm()
I_G_BD(graph, beta, D, n_samp = 1e4)
```

I_G_complete	<i>G Wishart normalising constant for complete graphs</i>
--------------	---

Description

This function returns the log of the transformed G-Wishart normalising constant $I_G(\text{beta}, D) = \int [S^p] \det(K)^{\text{beta}} \exp(-\text{tr}(KD)) dK$ for complete graphs G , using explicit formula $\det(D)^{-(\text{beta}+(p+1)/2)} \Gamma_n(\text{beta}+(p+1)/2)$

Usage

```
I_G_complete(p, beta, D = NULL, ratio = FALSE)
```

Arguments

<code>p</code>	A positive integer, indicating the number of vertices of the graph
<code>beta</code>	A constant > 0
<code>D</code>	A p by p real symmetric positive definite matrix (default, NULL, if for identity matrix)
<code>ratio</code>	Whether to output just the ratio compared to the value with I (default FALSE)

Value

A number $\log(I_G(\text{beta}, D))$

Examples

```
set.seed(42)
p <- 6
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
beta <- 1
I_G_complete(6, beta, D)
# compare with BDgraph gnorm()
adj <- matrix(1, nrow = 6, ncol = 6)
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_BD(graph, beta, D)
```

I_G_MC

*G Wishart normalising constant through MC integration***Description**

This function returns the log of transformed G-Wishart normalising constant $I_G(\beta, D) = \int [S^p_{++}(G)] \det(K)^\beta \exp(-\text{tr}(KD)) dK$ for connected graphs G . If there is no explicit formula, or the integral cannot be reduced into lower dimensions, MC integration is used.

Usage

```
I_G_MC(
  graph,
  beta,
  D,
  n_samp = 1000,
  nu = 10,
  err_flag = FALSE,
  int1d = TRUE,
  robust = FALSE,
  quench_k = 0,
  chordal = NULL
)
```

Arguments

graph	An igraph object, corresponding to a prime connected graph
beta	A constant > 0
D	A p by p real symmetric positive definite matrix
n_samp	number of sample points used in MC integration
nu	degree of freedom of the MC importance distribution (nu=0 is Gaussian)
err_flag	flag of whether to return the log relative error estimate (or not, default)
int1d	flag of whether to integrate numerically in 1d (default, MC integration otherwise)
robust	flag of whether to use robust means (default not to, but may offer numerical stability at the risk of bias)
quench_k	scaling factor of von Mises quenching (default not used, value of 0, but value around 1 may offer numerical stability at the risk of bias)
chordal	Optional. An igraph object, corresponding to a chordal completion of graph

Value

A number $\log(I_G(\beta, D))$

Examples

```

set.seed(42)
p <- 5
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
beta <- 1
adj <- matrix(0, nrow = p, ncol = p)
adj[1,5] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_MC(graph, beta, D)
# compare with BDgraph gnorm()
I_G_BD(graph, beta, D, n_samp = 1e5)

```

I_G_ratio_approx

*G Wishart normalising constant ratio approximation***Description**

This function returns the approximation of the ratio of log transformed G-Wishart normalising constants $I_G(\beta, D) / I_G(\beta, I)$ for graphs G . Graphs are decomposed into connected prime components. For each component if there is no explicit formula, the approximation is used. Note that this is the same as the ratio $C_G(\delta, D) / C_G(\delta, I)$ with $\delta = 2*\beta + 2$

Usage

```

I_G_ratio_approx(
  graph,
  beta,
  D,
  use_nlo = FALSE,
  connected = FALSE,
  prime = FALSE,
  chordal = NULL,
  use_comp = NULL
)

```

Arguments

graph	An igraph object, corresponding to a prime connected graph
beta	A constant > 0
D	A p by p real symmetric positive definite matrix
use_nlo	Whether to use the next-order approximation (TRUE) or not (default, FALSE). It is more accurate, but slower, and overwrites use_comp (as TRUE)
connected	Whether the graph is connected, if known (default FALSE)
prime	Whether the graph is prime, if known (default FALSE)
chordal	Optional. An igraph object, corresponding to a chordal completion of graph
use_comp	Whether to approximate using the graph (FALSE), the complement (TRUE) or the more efficient (default NULL)

Value

A number approximating $\log(I_G(\text{beta}, D)) - \log(I_G(\text{beta}, I))$

Examples

```
set.seed(42)
p <- 11
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
beta <- 1
graph <- igraph::make_graph(edges = c(1,2, 2,4, 2,5, 1,3, 3,5, # disconnected
                                     5,6, 6,7, 5,7, 4,7, # with prime decomposition too
                                     8,9, 8,11, 9,10, 10,11), n = 11, directed = FALSE)

I_G_ratio_approx(graph, beta, D)
I_G_ratio_approx(graph, beta, D, use_nlo = TRUE)
# compare with MC estimates
I_G_MC(graph, beta, D, n_samp = 1e5) - I_G_MC(graph, beta, diag(p), n_samp = 1e5)
```

I_G_ratio_approx_prime

G Wishart normalising constant ratio approximation for prime graphs

Description

This function returns the approximation of the ratio of log transformed G-Wishart normalising constants $I_G(\text{beta}, D) / I_G(\text{beta}, I)$ for connected prime graphs G . If there is no explicit formula, the approximation is used. Note that this is the same as the ratio $C_G(\text{delta}, D) / C_G(\text{delta}, I)$ with $\text{delta} = 2*\text{beta} + 2$

Usage

```
I_G_ratio_approx_prime(
  graph,
  beta,
  D,
  use_nlo = FALSE,
  chordal = NULL,
  use_comp = NULL
)
```

Arguments

graph	An igraph object, corresponding to a prime connected graph
beta	A constant > 0
D	A p by p real symmetric positive definite matrix
use_nlo	Whether to use the next-order approximation (TRUE) or not (default, FALSE). It is more accurate, but slower, and overwrites use_comp (as TRUE)

chordal	Optional. An igraph object, corresponding to a chordal completion of graph
use_comp	Whether to approximate using the graph (FALSE), the complement (TRUE) or the more efficient (default NULL)

Value

A number approximating $\log(I_G(\text{beta}, D)) - \log(I_G(\text{beta}, I))$

Examples

```
set.seed(42)
p <- 5
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
beta <- 1
adj <- matrix(0, nrow = p, ncol = p)
adj[1,5] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_ratio_approx_prime(graph, beta, D)
I_G_ratio_approx_prime(graph, beta, D, use_nlo = TRUE)
# compare with MC estimates
I_G_MC(graph, beta, D, n_samp = 1e5) - I_G_MC(graph, beta, diag(p), n_samp = 1e5)
```

I_G_special

G Wishart normalising constant for special cases

Description

This function returns the log of transformed G-Wishart normalising constant $I_G(\text{beta}, I) = \int [S^p_{++}(G)] \det(K)^\text{beta} * \exp(-\text{tr}(K)) dK$ for connected graphs G for special cases where there is an explicit formula, including involving a low-dimensional integral.

Usage

```
I_G_special(
  graph,
  beta,
  connected = FALSE,
  prime = FALSE,
  test_6_cycle = TRUE,
  test_k_partite = TRUE,
  chordal = NULL
)
```

Arguments

<code>graph</code>	An igraph object, corresponding to a prime connected graph
<code>beta</code>	A constant > 0
<code>connected</code>	Whether the graph is connected, if known (default FALSE)
<code>prime</code>	Whether the graph is prime, if known (default FALSE)
<code>test_6_cycle</code>	Flag whether to test whether the graph is a 6-cycle or its complement (default TRUE)
<code>test_k_partite</code>	Flag whether to test whether the graph is complete k partite (default TRUE)
<code>chordal</code>	Optional. An igraph object, corresponding to a chordal completion of graph

Value

A number $\log(I_G(\text{beta}, I))$

Examples

```

beta <- 1
p <- 5
adj <- matrix(0, nrow = p, ncol = p)
adj[1,5] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_special(graph, beta)
# compare with BDgraph gnorm()
I_G_BD(graph, beta, diag(p), n_samp = 1e5)
p <- 6
adj <- matrix(0, nrow = p, ncol = p)
adj[1,6] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- adj[5,6] <- 1 # 6-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_special(graph, beta)
# compare with BDgraph gnorm()
I_G_BD(graph, beta, diag(p), n_samp = 1e5)
graph <- igraph::complementer(graph) # complement of 6-cycle
I_G_special(graph, beta)
# compare with BDgraph gnorm()
I_G_BD(graph, beta, diag(p), n_samp = 1e5)
adj[upper.tri(adj)] <- 1
adj[2,3] <- adj[1,4] <- adj[5,6] <- 0 # complete graph with some edges missing
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
I_G_special(graph, beta)
# compare with BDgraph gnorm()
I_G_BD(graph, beta, diag(p), n_samp = 1e5)

```

I_Gnorm	<i>G Wishart normalising constant</i>
---------	---------------------------------------

Description

This function returns the log of transformed G-Wishart normalising constant $I_G(\beta, D) = \int [S^p_{++}(G)] \det(K)^\beta \exp(-\text{tr}(KD)) dK$ for graphs G . Graphs are decomposed into connected prime components. For each component if there is no explicit formula, or the integral cannot be reduced into one dimension, MC integration is used.

Usage

```
I_Gnorm(
  graph,
  beta,
  D,
  connected = FALSE,
  prime = FALSE,
  chordal = NULL,
  n_samp = 1000,
  nu = 10,
  robust = FALSE,
  quench_k = 0,
  err_flag = FALSE,
  useBDgraph = FALSE
)
```

Arguments

graph	An igraph object, corresponding to a prime connected graph
beta	A constant > 0
D	A p by p real symmetric positive definite matrix
connected	Whether the graph is connected, if known (default FALSE)
prime	Whether the graph is prime, if known (default FALSE)
chordal	Optional. An igraph object, corresponding to a chordal completion of graph
n_samp	number of sample points used in MC integration
nu	degree of freedom of the MC importance distribution (nu=0 is Gaussian)
robust	flag of whether to use robust means (default not to, but may offer numerical stability at the risk of bias)
quench_k	scaling factor of von Mises quenching (default value of 0 means not used, but value around 1 may offer numerical stability at the risk of bias)
err_flag	flag of whether to return the log relative error estimate (or not, default), only considered for prime connected graphs
useBDgraph	flag of whether to use BDgraph instead (or not, default), useful wrapper for comparisons

Value

A number $\log(I_G(\text{beta}, D))$

Examples

```
set.seed(42)
p <- 11
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
beta <- 1
graph <- igraph::make_graph(edges = c(1,2, 2,4, 2,5, 1,3, 3,5, # disconnected
                                     5,6, 6,7, 5,7, 4,7, # with prime decomposition too
                                     8,9, 8,11, 9,10, 10,11), n = 11, directed = FALSE)

I_Gnorm(graph, beta, D)
# compare with BDgraph
I_Gnorm(graph, beta, D, n_samp = 1e5, useBDgraph = TRUE)
```

I_GtoC_G

G-Wishart normalising constant reparametrisation one way

Description

This function returns the log of the G-Wishart normalising constant $\log(C_G(\text{delta}, D)) = \int [S^p_{++}(G)] \det(K)^{((\text{delta}-2)/2)} * \exp(-\text{tr}(KD)/2) dK$ from the transformed version $I_G(\text{beta}, D) = \int [S^p_{++}(G)] \det(K)^{\text{beta}} * \exp(-\text{tr}(KD)) dK$

Usage

```
I_GtoC_G(graph, IGval, beta)
```

Arguments

graph	An igraph object, corresponding to a prime connected graph
IGval	A number $\log(I_G(\text{beta}, D))$
beta	A constant > 0

Value

A number $\log(C_G(\text{delta}, D))$

is_6_cycle

*Determine whether the graph is the cycle of length 6 or its complement***Description**

Input an igraph object, this function tells you whether graph is the cycle of length 6, or its complement.

Usage

```
is_6_cycle(graph)
```

Arguments

graph An igraph object

Value

An integer, 1 means the graph is the cycle of length 6, 2 means the the graph is the complement of the cycle of length 6, 0 means otherwise

Examples

```
# C6 example 5 1 3 2 4 6
is_6_cycle(igraph::make_graph(edges = c(1,3, 1,5, 2,3, 2,4, 5,6, 6,4),
  n = 6, directed = FALSE))
# C6 complement example
is_6_cycle(igraph::make_graph(edges = c(1,2, 1,4, 1,6, 2,5, 2,6, 3,4,
  3,5, 3,6, 4,5), n = 6, directed = FALSE))
# not C6 examples
is_6_cycle(igraph::make_graph(edges = c(1,3, 1,5, 2,3, 2,4, 5,6, 6,4),
  n = 7, directed = FALSE))
is_6_cycle(igraph::make_graph(edges = c(1,4, 1,5, 2,3, 2,4, 5,6, 6,4),
  n = 6, directed = FALSE))
```

is_k_partite

*Determine whether the graph is complete k partite***Description**

Input an igraph object, this function tells you the size of the partition if the graph is a k partite graph.

Usage

```
is_k_partite(graph)
```

Arguments

graph An igraph object

Value

A vector of the size of the partition. The length of the vector is k. Empty vector means that the graph is not a complete k partite graph.

Examples

```
# not complete k-partite example
is_k_partite(igraph::make_graph(edges = c(1,2, 2,4, 2,5, 1,3, 3,5, 5,6, 6,7,
5,7, 4,7), n = 7, directed = FALSE))
# complete 4-partite example, (1,5), (2,4), (3,7), (6)
is_k_partite(igraph::make_graph(edges = c(1,2, 1,3, 1,4, 1,6, 1,7, 2,3, 2,5, 2,6,
2,7, 3,4, 3,5, 3,6, 4,5, 4,6, 4,7, 5,6, 5,7, 6,7), n = 7, directed = FALSE))
```

Iss_cmat

Isserlis complement matrix

Description

This function returns the Isserlis matrix (rows/columns arranged in some order) with respect to a matrix and a graph for the missing edges (complement of the graph).

Usage

```
Iss_cmat(M, graph)
```

Arguments

M An p by p matrix

graph An igraph object, corresponding to the graph

Value

An m by m matrix, where m is the number of missing edges in the graph

Examples

```
# example code
adj <- matrix(0, nrow = 5, ncol = 5)
adj[1,5] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
Iss_cmat(matrix(1:25, nrow = 5, byrow = TRUE), graph)
```

Iss_mat	<i>Isserlis matrix</i>
---------	------------------------

Description

This function returns the Isserlis matrix (rows/columns arranged in some order) with respect to a matrix and a graph

Usage

```
Iss_mat(M, graph)
```

Arguments

M	An p by p matrix
graph	An igraph object, corresponding to the graph

Value

An m by m matrix, where m is the number of vertices and edges in the graph

Examples

```
# example code
adj <- matrix(0, nrow = 5, ncol = 5)
adj[1,5] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
Iss_mat(matrix(1:25, nrow = 5, byrow = TRUE), graph)
```

PD_complete	<i>PD-completion</i>
-------------	----------------------

Description

This function returns the PD-completion of a matrix with respect to a graph. The entries of the given matrix corresponding to the missing edges are modified such that the inverse has zeros in the places of missing edges.

Usage

```
PD_complete(graph, D, missing_edges = NULL, tol = 1e-12)
```

Arguments

graph	An igraph object, corresponding to the graph
D	A real symmetric positive definite p by p matrix
missing_edges	optional argument of edges to complete
tol	A tolerance to stop the numerical iterations

Value

A real symmetric positive definite p by p matrix

Examples

```
set.seed(42)
p <- 5
data <- matrix(runif(10*p), ncol = p)
D <- t(data) %*% data
adj <- matrix(0, nrow = p, ncol = p)
adj[1,5] <- adj[1,2] <- adj[2,3] <- adj[3,4] <- adj[4,5] <- 1 # 5-cycle
graph <- igraph::graph_from_adjacency_matrix(adj, mode = c("max"))
(D_tilde <- PD_complete(graph, D))
(round(solve(D_tilde), 6)) # has zeros at the missing edges
```

prime_decomp

Prime Decomposition

Description

This function returns the prime components and the minimal separators of a connected graph.

Usage

```
prime_decomp(graph)
```

Arguments

graph	An igraph object, a connected graph
-------	-------------------------------------

Value

A list of the prime components and the separators

Examples

```
graph <- igraph::make_graph(edges = c(1,2, 2,4, 2,5, 1,3, 3,5,
  5,6, 6,7, 5,7, 4,7), n = 7, directed = FALSE)
prime_decomp(graph)
```

Index

C_GtoI_G, [2](#)
check_prime_connected, [3](#)

I_G_BD, [3](#)
I_G_chordal, [4](#)
I_G_complete, [5](#)
I_G_MC, [6](#)
I_G_ratio_approx, [7](#)
I_G_ratio_approx_prime, [8](#)
I_G_special, [9](#)
I_Gnorm, [11](#)
I_GtoC_G, [12](#)
is_6_cycle, [13](#)
is_k_partite, [13](#)
Iss_cmat, [14](#)
Iss_mat, [15](#)

PD_complete, [15](#)
prime_decomp, [16](#)