

Package: rootWishartHD (via r-universe)

July 16, 2026

Type Package

Title Exact and Log-Scale Tail Probabilities for Roy's Largest Root

Version 0.95.2

Description Provides distribution functions and log-scale tail probabilities for Roy's largest root in single and double Wishart (Jacobi ensemble) settings. This package is derived from the 'rootWishart' package by Maxime Turgeon and extends it with numerically robust log-CDF/log-survival evaluation, tail-aware adaptive precision, and high-dimensional validation utilities, based on Chiani (2014) [<DOI:10.1016/j.jmva.2014.04.002>](https://doi.org/10.1016/j.jmva.2014.04.002) and Chiani (2016) [<DOI:10.1016/j.jmva.2015.10.007>](https://doi.org/10.1016/j.jmva.2015.10.007).

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation yes

LinkingTo Rcpp, RcppEigen, BH

Imports Rcpp

Suggests tinytest, knitr, rmarkdown, rWishart, corpcor

VignetteBuilder knitr

URL <https://github.com/stepanvl/rootWishartHD>

BugReports <https://github.com/stepanvl/rootWishartHD/issues>

Config/roxygen2/version 8.0.0

Author Stepan Grinek [aut, cre], Maxime Turgeon [ctb] (Original rootWishart package codebase)

Maintainer Stepan Grinek <stepan.grinek@jax.org>

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

Date/Publication 2026-07-16 12:40:02 UTC

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

RemoteRef HEAD

RemoteSha f3a383ccc69885df986597d3551e75dd75d7d529

Contents

doubleWishart	2
doubleWishart_log	3
doubleWishart_pvalue	4
doubleWishart_qalpha	5
doubleWishart_robustPfaffians	6
doubleWishart_scaled	6
rootWishartHD_mpfr_enabled	7
singleWishart_cdf	8
singleWishart_log	9
singleWishart_pvalue	10
singleWishart_qalpha	10
Index	12

doubleWishart	<i>Double-Wishart largest-root CDF</i>
---------------	--

Description

Computes the CDF $F(\theta)$ for Roy's largest root in the real double-Wishart/Jacobi ensemble.

Usage

```
doubleWishart(x, s = NULL, m, n,
  type = c("double", "arbitrary", "fixed"), verbose = TRUE,
  force_multiprecision = getOption("rootWishartHD.force_multiprecision", FALSE),
  force_mpfr = NULL, p = NULL)
```

Arguments

x	Numeric vector of evaluation points on the Jacobi scale $\theta \in [0, 1]$. For beta type II roots λ , use $\theta = \lambda/(1 + \lambda)$.
s	Integer matrix dimension in Chiani's parameterization.
p	Optional alias for s, retained for older scripts.
m, n	Shape/ensemble parameters in Chiani's Pfaffian formulation.
type	"double" for fast double precision or "arbitrary" for multiprecision. "fixed" is a legacy alias for "double".
verbose	Logical; if TRUE, prints backend messages.
force_multiprecision	Logical; if TRUE, uses the arbitrary-precision backend even when type="double". Can also be set globally with options(rootWishartHD.force_multiprecision = TRUE).
force_mpfr	Deprecated alias for force_multiprecision. If MPFR is explicitly requested but the package was built with DW_USE_MPFR=0, a warning is issued and Boost cpp_dec_float is used instead.

Details

The CRAN-safe default multiprecision backend is Boost's header-only `cpp_dec_float` from **BH** (compiled with `DW_USE_MPFR=0`). Local source builds can opt in to MPFR/GMP by installing those libraries and using `DW_USE_MPFR=1` R CMD INSTALL `rootWishartHD_*.tar.gz`. See `rootWishartHD_mpfr_enabled()` and the README.

Value

Numeric vector of CDF values in the same order as `x`.

Examples

```
x <- seq(0.1, 0.9, length.out = 5)
doubleWishart(x, s = 2, m = 6, n = 9, type = "double", verbose = FALSE)
```

<code>doubleWishart_log</code>	<i>Double-Wishart log-CDF and log-survival</i>
--------------------------------	--

Description

Returns log-scale probabilities for the double-Wishart largest root, avoiding 0/1 saturation in extreme tails.

Usage

```
doubleWishart_log(x, s = NULL, m, n,
  type = c("double", "arbitrary", "fixed"), tail = c("lower", "upper"),
  verbose = TRUE,
  force_multiprecision = getOption("rootWishartHD.force_multiprecision", FALSE),
  force_mpfr = NULL, direct_b = FALSE, scale_iter = 5L,
  pf_method = c("auto", "gauss", "svd", "schur", "lu"),
  adaptive = FALSE, start_digits10 = 300L, max_digits10 = 20000L,
  tol = 1e-30, use_arb = FALSE, p = NULL)
```

Arguments

`x` Numeric vector on the Jacobi $\theta \in [0, 1]$ scale.

`s, p, m, n, type, verbose, force_multiprecision, force_mpfr`
See `doubleWishart`.

`tail` "lower" returns $\log F(\theta)$; "upper" returns $\log(1 - F(\theta))$.

`direct_b, scale_iter, pf_method, adaptive, start_digits10, max_digits10, tol, use_arb`
Numerical controls; see `doubleWishart_scaled`.

Details

Runtime adaptive precision (adaptive=TRUE) requires an MPFR/GMP build. With the default DW_USE_MPFR=0 build, adaptive requests are downgraded to fixed Boost cpp_dec_float precision with a warning.

When type="arbitrary", rootWishartHD uses Boost cpp_dec_float by default. MPFR/GMP is optional and requires building with DW_USE_MPFR=1; rootWishartHD_mpf_enabled() reports which backend was compiled.

Value

Numeric vector of log probabilities in the same order as x.

Examples

```
doubleWishart_log(c(0.8, 0.9), s = 5, m = 10, n = 10,
  tail = "upper", verbose = FALSE)
```

doubleWishart_pvalue	<i>Double-Wishart upper-tail p-values</i>
----------------------	---

Description

Computes upper-tail probabilities $P(\Theta \geq \theta)$ using log-survival when needed.

Usage

```
doubleWishart_pvalue(lambda_or_theta, s, m, n,
  input = c("auto", "theta", "lambda"),
  scale = c("p", "logp", "log10p"),
  backend = c("logexp", "robust"), type = "arbitrary",
  adaptive = rootWishartHD_mpf_enabled(), start_digits10 = 300L, max_digits10 = 20000L,
  tol = 1e-12, pass1_max_digits10 = 3000L,
  fast_try_double = TRUE, fast_backend = c("scaled", "base"),
  verbose = FALSE)
```

Arguments

lambda_or_theta	Observed largest-root values.
s, m, n	Jacobi parameters in Chiani's parameterization.
input	"theta", "lambda", or "auto".
scale	Return ordinary p-values, natural log p-values, or $-\log_{10}$ p-values.
backend	"logexp" uses doubleWishart_log; "robust" uses doubleWishart_robustPfaffians.
type, adaptive, start_digits10, max_digits10, tol	Precision controls.

pass1_max_digits10 Digit cap for the first exact pass. Only unresolved points are refined.

fast_try_double Try a fast double-precision CDF pass before exact log-tail evaluation.

fast_backend "scaled" uses doubleWishart_scaled; "base" uses doubleWishart.

verbose Logical.

Value

Numeric vector with status and method attributes.

doubleWishart_qalpha *Double-Wishart critical values*

Description

Finds the largest-root critical value with $P(\Theta \geq \theta) = \alpha$.

Usage

```
doubleWishart_qalpha(alpha, s, m, n,
  return = c("theta", "lambda"), backend = c("logexp", "robust"),
  type = "arbitrary", adaptive = rootWishartHD_mpfr_enabled(), start_digits10 = 300L,
  max_digits10 = 20000L, tol = 1e-12, pass1_max_digits10 = 3000L,
  fast_try_double = TRUE, fast_backend = c("scaled", "base"),
  verbose = FALSE, max_iter = 120L)
```

Arguments

alpha Upper-tail probability in $(0, 1)$.

s, m, n Jacobi parameters in Chiani's parameterization.

return Return "theta" or beta type II "lambda".

backend, type, adaptive, start_digits10, max_digits10, tol,
pass1_max_digits10, fast_try_double, fast_backend, verbose See doubleWishart_pvalue.

max_iter Maximum number of bisection iterations.

Value

Numeric scalar with status and method attributes.

doubleWishart_robustPfaffians

Robust double-Wishart log-tail dispatcher

Description

A robust log-scale dispatcher for large or difficult double-Wishart cases. It uses conservative Pfaffian settings and adaptive multiprecision by default.

Usage

```
doubleWishart_robustPfaffians(x, s = NULL, m, n,
  type = c("double", "arbitrary", "fixed"), tail = c("lower", "upper"),
  verbose = TRUE,
  force_multiprecision = getOption("rootWishartHD.force_multiprecision", FALSE),
  force_mpfr = NULL, direct_b = FALSE, scale_iter = 8L,
  adaptive = rootWishartHD_mpfr_enabled(), start_digits10 = 300L, max_digits10 = 20000L,
  tol = 1e-30, use_arb = FALSE, p = NULL)
```

Arguments

x Numeric vector on the Jacobi $\theta \in [0, 1]$ scale.

s, p, m, n, type, verbose, force_multiprecision, force_mpfr
 See doubleWishart.

tail "lower" for log-CDF or "upper" for log-survival.

direct_b, scale_iter, adaptive, start_digits10, max_digits10, tol,
use_arb
 Numerical controls; see doubleWishart_log.

Value

Numeric vector of log probabilities.

doubleWishart_scaled *Scaled double-Wishart CDF backend*

Description

Computes a double-valued CDF using the scaled Pfaffian backend. This is the backend used by doubleWishart() for multiprecision or large s cases.

Usage

```
doubleWishart_scaled(x, s = NULL, m, n,
  type = c("double", "arbitrary", "fixed"), verbose = TRUE,
  force_multiprecision = getOption("rootWishartHD.force_multiprecision", FALSE),
  force_mpfr = NULL, direct_b = FALSE, scale_iter = 5L,
  pf_method = c("auto", "gauss", "svd", "schur", "lu"),
  adaptive = FALSE, start_digits10 = 200L, max_digits10 = 5000L,
  tol = 1e-30, use_arb = FALSE, p = NULL)
```

Arguments

x Numeric vector on the Jacobi $\theta \in [0, 1]$ scale.

s, p, m, n, type, verbose, force_multiprecision, force_mpfr
See doubleWishart.

direct_b Use the non-recursive construction for intermediate incomplete-beta terms.

scale_iter Number of symmetric equilibration iterations before Pfaffian evaluation.

pf_method Pfaffian backend. "auto" tries robust choices.

adaptive, start_digits10, max_digits10, tol
Adaptive multiprecision controls.

use_arb Reserved for ARB-enabled local builds.

Details

Runtime adaptive precision (adaptive=TRUE) requires an MPFR/GMP build. With the default DW_USE_MPFR=0 build, adaptive requests are downgraded to fixed Boost cpp_dec_float precision with a warning.

For extreme upper tails, prefer doubleWishart_log(tail="upper"); any double-valued CDF may saturate to 1.

Value

Numeric vector of CDF values.

Examples

```
doubleWishart_scaled(c(0.5, 0.8), s = 5, m = 10, n = 10, verbose = FALSE)
```

```
rootWishartHD_mpfr_enabled
```

Check whether MPFR/GMP support was compiled

Description

Reports whether the package was built with the optional MPFR/GMP backend.

Usage

```
rootWishartHD_mpfr_enabled()
```

Details

The default CRAN-safe build uses `DW_USE_MPFR=0`, which selects Boost's header-only `cpp_dec_float` backend from **BH**. To enable MPFR/GMP in a local source build, install MPFR and GMP and run, for example, `DW_USE_MPFR=1 R CMD INSTALL rootWishartHD_*.tar.gz`. This is optional; arbitrary precision still works with `cpp_dec_float`.

Value

Logical scalar.

Examples

```
rootWishartHD_mpfr_enabled()
```

singleWishart_cdf	<i>Single-Wishart largest-eigenvalue CDF</i>
-------------------	--

Description

Computes the CDF of the largest eigenvalue of a real single-Wishart matrix.

Usage

```
singleWishart_cdf(x, n_min, n_max,
  type = c("double", "arbitrary", "fixed"), verbose = TRUE,
  force_multiprecision = getOption("rootWishartHD.force_multiprecision", FALSE),
  force_mpfr = NULL, adaptive = FALSE, start_digits10 = 200L,
  max_digits10 = 5000L, tol = 1e-12)
```

Arguments

x	Numeric vector of nonnegative evaluation points on the eigenvalue scale.
n_min, n_max	Positive integer single-Wishart dimensions with <code>n_max >= n_min</code> .
type, verbose, force_multiprecision, force_mpfr	See <code>doubleWishart</code> .
adaptive, start_digits10, max_digits10, tol	Adaptive multiprecision controls.

Details

Runtime adaptive precision (`adaptive=TRUE`) requires an MPFR/GMP build. With the default `DW_USE_MPFR=0` build, adaptive requests are downgraded to fixed Boost `cpp_dec_float` precision with a warning.

Value

Numeric vector of CDF values.

Examples

```
singleWishart_cdf(c(10, 20), n_min = 5, n_max = 10, verbose = FALSE)
```

singleWishart_log	<i>Single-Wishart log-CDF and log-survival</i>
-------------------	--

Description

Returns log-scale probabilities for the single-Wishart largest eigenvalue.

Usage

```
singleWishart_log(x, n_min, n_max,
  type = c("double", "arbitrary", "fixed"), tail = c("lower", "upper"),
  verbose = TRUE,
  force_multiprecision = getOption("rootWishartHD.force_multiprecision", FALSE),
  force_mpfr = NULL, adaptive = FALSE, start_digits10 = 300L,
  max_digits10 = 20000L, tol = 1e-12)
```

Arguments

x	Numeric vector of nonnegative evaluation points on the eigenvalue scale.
n_min, n_max	Positive integer single-Wishart dimensions with n_max >= n_min.
tail	"lower" returns log-CDF; "upper" returns log-survival.
type, verbose, force_multiprecision, force_mpfr	See doubleWishart.
adaptive, start_digits10, max_digits10, tol	Adaptive multiprecision controls.

Details

Runtime adaptive precision (adaptive=TRUE) requires an MPFR/GMP build. With the default DW_USE_MPFR=0 build, adaptive requests are downgraded to fixed Boost cpp_dec_float precision with a warning.

Value

Numeric vector of log probabilities.

Examples

```
singleWishart_log(c(20, 30), n_min = 5, n_max = 10,
  tail = "upper", verbose = FALSE)
```

singleWishart_pvalue *Single-Wishart upper-tail p-values*

Description

Computes $P(X \geq x)$ for the single-Wishart largest eigenvalue.

Usage

```
singleWishart_pvalue(x, n_min, n_max,
  scale = c("p", "logp", "log10p"), backend = c("logexp"),
  adaptive = rootWishartHD_mpfr_enabled(), start_digits10 = 300L, max_digits10 = 20000L,
  tol = 1e-12, pass1_max_digits10 = 3000L,
  fast_try_double = TRUE, verbose = FALSE, sanity_check = TRUE,
  sanity_ratio = 1e6, sanity_min_cdf = 1e-12)
```

Arguments

x	Observed largest eigenvalue values on the single-Wishart scale.
n_min, n_max	Positive integer single-Wishart dimensions.
scale	Return ordinary p-values, natural log p-values, or $-\log_{10}$ p-values.
backend	Currently "logexp", using singleWishart_log.
adaptive, start_digits10, max_digits10, tol, pass1_max_digits10	Precision controls.
fast_try_double	Try singleWishart_cdf first before exact log-tail evaluation.
verbose, sanity_check, sanity_ratio, sanity_min_cdf	Diagnostic and fast-path sanity controls.

Value

Numeric vector with status and method attributes.

singleWishart_qalpha *Single-Wishart critical values*

Description

Finds x such that $P(X \geq x) = \alpha$.

Usage

```
singleWishart_qalpha(alpha, n_min, n_max, return = c("x"),
  backend = c("logexp"), adaptive = rootWisharHD_mpfr_enabled(), start_digits10 = 300L,
  max_digits10 = 20000L, tol = 1e-12, pass1_max_digits10 = 3000L,
  fast_try_double = TRUE, verbose = FALSE, max_iter = 120L,
  max_expand = 200L)
```

Arguments

alpha	Upper-tail probability in $(0, 1)$.
n_min, n_max	Positive integer single-Wishart dimensions.
return	Only "x"; returns the single-Wishart eigenvalue scale.
backend, adaptive, start_digits10, max_digits10, tol,	
pass1_max_digits10, fast_try_double, verbose	See singleWishart_pvalue.
max_iter, max_expand	Bisection and bracket-expansion controls.

Value

Numeric scalar with status and method attributes.

Index

- * **distribution**

- doubleWishart, [2](#)

- * **multivariate**

- doubleWishart, [2](#)

doubleWishart, [2](#)

doubleWishart_log, [3](#)

doubleWishart_pvalue, [4](#)

doubleWishart_qalpha, [5](#)

doubleWishart_robustPfaffians, [6](#)

doubleWishart_scaled, [6](#)

rootWisharHD_mpfr_enabled, [7](#)

singleWishart_cdf, [8](#)

singleWishart_log, [9](#)

singleWishart_pvalue, [10](#)

singleWishart_qalpha, [10](#)