

ghkfast() — GHK multivariate normal simulator using pregenerated points

Description Diagnostics	Syntax Also see	Remarks and examples	Conformability
----------------------------	--------------------	----------------------	----------------

Description

Please see [M-5] [ghk\(\)](#). The routines documented here do the same thing, but `ghkfast()` can be faster at the expense of using more memory. First, code `S = ghkfast_init(...)` and then use `ghkfast(S, ...)` to obtain the simulated values. There is a time savings because the simulation points are generated once in `ghkfast_init()`, whereas for `ghk()` the points are generated on each call to `ghk()`. Also, `ghkfast()` can generate simulated probabilities from the generalized Halton sequence; see [M-5] [halton\(\)](#).

`ghkfast_init(n, npts, dim, method)` computes the simulation points to be used by `ghkfast()`. Inputs `n`, `npts`, and `dim` are the number of observations, the number of repetitions for the simulation, and the maximum dimension of the multivariate normal (MVN) distribution, respectively. Input `method` specifies the type of points to generate and can be one of "halton", "hammersley", "random", or "ghalton".

`ghkfast(S, X, V)` returns an $n \times 1$ real vector containing the simulated values of the MVN distribution with $dim \times dim$ variance-covariance matrix V at the points stored in the rows of the $n \times dim$ matrix X .

`ghkfast(S, X, V, dfdx, dfdv)` does the same thing as `ghkfast(S, X, V)` but also returns the first-order derivatives of the simulated probability with respect to the rows of X in `dfdx` and the simulated probability derivatives with respect to `vech(V)` in `dfdv`. See `vech()` in [M-5] [vec\(\)](#) for details of the half-vectorized operator.

The `ghk_query_n(S)`, `ghk_query_npts(S)`, `ghk_query_dim(S)`, and `ghk_query_method(S)` functions extract the number of observations, number of simulation points, maximum dimension, and method of point-set generation that is specified in the construction of the transmorphic object S . Use `ghk_query_rseed(S)` to retrieve the uniform random-variate seed used to generate the "random" or "ghalton" point sets. The `ghkfast_query_pointset_i(S, i)` function will retrieve the i th point set used to simulate the MVN probability for the i th observation.

The `ghkfast_i(S, X, V, i, ...)` function computes the probability and derivatives for the i th observation, $i = 1, \dots, n$.

Syntax

$S = \text{ghkfast_init}(\text{real scalar } n, \text{npts}, \text{dim}, \text{string scalar method})$

(varies) `ghkfast_init_pivot(S [, real scalar pivot])`

(varies) `ghkfast_init_antithetics(S [, real scalar anti])`

real scalar `ghkfast_query_n(S)`

real scalar `ghkfast_query_npts(S)`

real scalar `ghkfast_query_dim(S)`

string scalar `ghkfast_query_method(S)`

string scalar `ghkfast_query_rseed(S)`

real matrix `ghkfast_query_pointset_i(S, i)`

real colvector `ghkfast(S, real matrix X, V)`

real colvector `ghkfast(S, real matrix X, V, dfdx, dfdv)`

real scalar `ghkfast_i(S, real matrix X, V, i)`

real scalar `ghkfast_i(S, real matrix X, V, i, dfdx, dfdv)`

where S , if it is declared, should be declared

transmorphic S

and where *method* specified in `ghkfast_init()` is

<i>method</i>	Description
"halton"	Halton sequences
"hammersley"	Hammersley's variation of the Halton set
"random"	pseudorandom uniforms
"ghalton"	generalized Halton sequences

Remarks and examples

[stata.com](http://www.stata.com)

For problems where repetitive calls to the GHK algorithm are required, `ghkfast()` might be a preferred alternative to `ghk()`. Generating the points once at the outset of a program produces a speed increase. For problems with many observations or many simulation points per observation, `ghkfast()` will be faster than `ghk()` at the cost of requiring more memory.

If `ghkfast()` is used within a likelihood evaluator for `ml` or `optimize()`, you will need to store the transmorphic object S as an [external](#) global and reuse the object with each likelihood evaluation. Alternatively, the initialization function for `optimize()`, `optimize_init_argument()`, can be used.

Prior to calling `ghkfast()`, call `ghkfast_init_npivot(S, 1)` to turn off the integration interval pivoting that takes place in `ghkfast()`. By default, `ghkfast()` pivots the wider intervals of integration (and associated rows/columns of the covariance matrix) to the interior of the multivariate integration to improve quadrature accuracy. This option may be useful when `ghkfast()` is used in a likelihood evaluator for [R] `ml` or [M-5] `optimize()` and few simulation points are used for each observation. Here the pivoting may cause discontinuities when computing numerical second-order derivatives using finite differencing (for the Newton–Raphson technique), resulting in a non–positive-definite Hessian.

Also the sequences "halton", "hammersley", and "random", `ghkfast()` will use the generalized Halton sequence, "ghalton". Generalized Halton sequences have the same uniform coverage (low discrepancy) as the Halton sequences with the addition of a pseudorandom uniform component. Therefore, "ghalton" sequences are like "random" sequences in that you should set the random-number seed before using them if you wish to replicate the same point set; see [M-5] `runiform()`.

Conformability

All initialization functions have 1×1 inputs and have 1×1 or *void* outputs, and all query functions have the *transmorphic* input and 1×1 outputs except

`ghkfast_init(n, npts, dim, method)`:

input:

n: 1×1
npts: 1×1
dim: 1×1
method: 1×1

output:

result: *transmorphic*

`ghkfast_query_pointset_i(S, i)`:

input:

S: *transmorphic*
i: 1×1

output:

result: $npts \times dim$

`ghkfast(S, X, V)`:

input:

S: *transmorphic*
X: $n \times dim$
V: $dim \times dim$ (symmetric, positive definite)

output:

result: $n \times 1$

`ghkfast(S, X, V, dfdx, dfdv)`:

input:

S: *transmorphic*
X: $n \times dim$
V: $dim \times dim$ (symmetric, positive definite)

output:

result: $n \times 1$
dfdx: $n \times dim$
dfdv: $n \times dim(dim + 1)/2$

`ghkfast_i(S, X, V, i, dfdx, dfdv)`:

input:

S: *transmorphic*
X: $n \times \text{dim}$ or $1 \times \text{dim}$
V: $\text{dim} \times \text{dim}$ (symmetric, positive definite)
i: 1×1 ($1 \leq i \leq n$)

output:

result: $n \times 1$
dfdx: $1 \times \text{dim}$
dfdv: $1 \times \text{dim}(\text{dim} + 1)/2$

Diagnostics

`ghkfast_init(n, npts, dim, method)` aborts with error if the dimension, *dim*, is greater than 20.

`ghkfast(S, X, V, ...)` and `ghkfast_i(S, X, V, i, ...)` require that *V* be symmetric and positive definite. If *V* is not positive definite, then the returned vector (scalar) is filled with missings.

Also see

[M-5] `ghk()` — Geweke–Hajivassiliou–Keane (GHK) multivariate normal simulator

[M-5] `halton()` — Generate a Halton or Hammersley set

[M-4] `statistical` — Statistical functions