Title

stata.com

ghk( ) — Geweke–Hajivassiliou–Keane (GHK) multivariate normal simulator

Description

The `ghk*()` set of functions provide a Geweke–Hajivassiliou–Keane (GHK) multivariate normal simulator.

S = `ghk_init(`*npts*`)` initializes the simulator with the desired number of simulation points and returns a transmorphic object *S*, which is a handle that should be used in subsequent calls to other `ghk*()` functions. Calls to `ghk_init_method(`*S*`,` *method*`)`, `ghk_init_start(`*S*`,` *start*`)`, `ghk_init_pivot(`*S*`,` *pivot*`)`, and `ghk_init_antithetics(`*S*`,` *anti*`)` prior to calling `ghk(`*S*`, ...)` allow you to modify the simulation algorithm through the object *S*.

`ghk(`*S*`,` *x*`,` *V*`)` returns a real scalar containing the simulated value of the multivariate normal (MVN) distribution with variance–covariance *V* at the point *x*. First, code *S* = `ghk_init(`*npts*`)` and then use `ghk(`*S*`, ...)` to obtain the simulated value based on *npts* simulation points.

`ghk(`*S*`,` *x*`,` *V*`,` *dfdx*`,` *dfdv*`)` does the same thing but also returns the first-order derivatives of the simulated probability with respect to *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_npts(`*S*`)` function returns the number of simulation points, the same value given in the construction of the transmorphic object *S*.

Syntax

S = `ghk_init(`*real scalar npts*`)`

(varies) `ghk_init_method(`*S* [`,` *string scalar method*]`)`

(varies) `ghk_init_start(`*S* [`,` *real scalar start*]`)`

(varies) `ghk_init_pivot(`*S* [`,` *real scalar pivot*]`)`

(varies) `ghk_init_antithetics(`*S* [`,` *real scalar anti*]`)`

real scalar `ghk_query_npts(`*S*`)`

real scalar `ghk(`*S*`,` *real vector x*`,` *V*`)`

real scalar `ghk(`*S*`,` *real vector x*`,` *V*`,` *real rowvector dfdx*`,` *dfdv*`)`

where *S*, if declared, should be declared

`transmorphic` *S*

and where *method*, optionally specified in `ghk_init_method()`, is

method	Description
`"halton"`	Halton sequences
`"hammersley"`	Hammersley's variation of the Halton set
`"random"`	pseudorandom uniforms

Remarks and examples

stata.com

Halton and Hammersley point sets are composed of deterministic sequences on [0,1] and, for sets of dimension less than 10, generally have better coverage than that of the uniform pseudorandom sequences.

Antithetic draws effectively double the number of points and reduce the variability of the simulated probability. For draw u, the antithetic draw is $1 - u$. To use antithetic draws, call `ghk_init_antithetic(`*S*`, 1)` prior to executing `ghk()`.

By default, `ghk()` will pivot the wider intervals of integration (and associated rows/columns of the covariance matrix) to the interior of the multivariate integration. This improves the accuracy of the quadrature estimate. When `ghk()` is used in a likelihood evaluator for [R] **ml** or [M-5] **optimize( )**, discontinuities may result in the computation of numerical second-order derivatives using finite differencing (for the Newton–Raphson optimize technique) when few simulation points are used, resulting in a non–positive-definite Hessian. To turn off the interval pivoting, call `ghk_init_pivot(`*S*`, 0)` prior to executing `ghk()`.

If you are using `ghk()` in a likelihood evaluator, be sure to use the same sequence with each call to the likelihood evaluator. For a uniform pseudorandom sequence, `ghk_init_method("random")`, set the seed of the uniform random-variate generator—see `rseed()` in [M-5] **runiform( )**—to the same value with each call to the likelihood evaluator.

If you are using the Halton or Hammersley point sets, you will want to keep the sequences going with each call to `ghk()` within one likelihood evaluation. This can be done in one expression executed after each call to `ghk()`: `ghk_init_start(`*S*`, ghk_init_start(`*S*`) + ghk_query_npts(`*S*`))`. With each call to the likelihood evaluator, you will need to reset the starting index to 1. This last point assumes that the transmorphic object *S* is not re-created on each call to the likelihood evaluator.

Unlike `ghkfast_init()` (see [M-5] **ghkfast( )**), the transmorphic object *S* created by `ghk_init()` is inexpensive to create, so it is possible to re-create it with each call to your likelihood evaluator instead of storing it as external global and reusing the object with each likelihood evaluation. Alternatively, the initialization function for `optimize()`, `optimize_init_arguments()`, can be used.

Conformability

All initialization functions have 1×1 inputs and have 1×1 or *void* outputs except

`ghk_init(`*npts*`)`:
input:
npts: 1×1
output:
S: transmorphic

`ghk_query_npts(`*S*`)`:
input:
S: transmorphic
output:
result: 1×1

`ghk(`*S*`, `*x*`, `*V*`)`:
input:
S: transmorphic
x: $1 \times m$ or $m \times 1$
V: $m \times m$ (symmetric, positive definite)
output:
result: 1×1

`ghk(`*S*`, `*x*`, `*V*`, `*dfdx*`, `*dfdv*`)`:
input:
S: transmorphic
x: $1 \times m$ or $m \times 1$
V: $m \times m$ (symmetric, positive definite)
output:
result: 1×1
dfdx: $1 \times m$
dfdv: $1 \times m(m+1)/2$

Diagnostics

The maximum dimension, m, is 20.

V must be symmetric and positive definite. `ghk()` will return a missing value when V is not positive definite. When `ghk()` is used in an `ml` (or `optimize()`) likelihood evaluator, return a missing likelihood to `ml` and let `ml` take the appropriate action (that is, step halving).

Also see

[M-5] **ghkfast( )** — GHK multivariate normal simulator using pregenerated points

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

[M-4] **statistical** — Statistical functions