Package: hashprng (via r-universe)

July 11, 2024

Type Package

Title Hash-Based Matching Pseudo-Random Number Generation

Version 0.2.0.1000

Description Provides helper functions for use of hash-based matching (HBM) for pseudo-random number generation (PRNG) in stochastic simulations. HBM-PRNG is an approach to simplify matching synthetic experiment samples, which ensures that matched runs different only in the focal parameters, not in their chance events.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LinkingTo Rcpp

Imports Rcpp

Suggests data.table, ggplot2, hexSticker, knitr, parallel, rmarkdown, roxygen2, spelling, sysfonts, testthat (>= 3.0.0), usethis

Config/Needs/website r-lib/pkgdown

Config/Needs/hexsticker hexSticker, sysfonts

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

Repository https://epinowcast.r-universe.dev

RemoteUrl https://github.com/epinowcast/hashprng

RemoteRef v0.2.0

RemoteSha 3c8c2ea5bc512d2ae8c79a26d1135e34b36a7ad1

Contents

`hashprng` *hashprng: Hash-Based Matching Pseudo-Random Number Generation*

Description

The hashprng package provides single function for use during stochastic simulation to streamline salting + event hashing + reseeding the R random number generator. The sole function in the package is `hash_seed`.

`hash_seed` *Hash-Based Matching Pseudo-Random Number Generation*

Description

Hash-Based Matching Pseudo-Random Number Generation

Usage

```
hash_seed(salt, ...)

hash_salt(salt, ...)
```

Arguments

`salt`	the matching value for a particular collection of simulations
`...`	distinguishing features to identify the event; see details.

Details

These functions provide convenient invocation for hash-based matching pseudo-random generation (HBM-PRNG).

`hash_seed` uses a `salt` value along with distinguishing features of an event. Typically, `salt` distinguishes an overall sample simulation, but it can also be a temporarily computed value for events that share some-but-not all features.

`hash_salt` computes a *partial* hash, for when several events need draws, but share a partially consistent feature set. The result of `hash_salt` can for the consistent features can be computed once, then provided to `hash_seed` along with remaining distinct features.

For matched stochastic simulation, we desire a few properties:

- the *same* random events are resolved *consistently*
- possibility of different stochastic samples
- reproducibility of pseudo-random simulations

Traditional PRNG seeding provides the latter points. To the extent that the PRNG is traversed the same way across simulations, events will also be resolved consistently. However, once event resolution leads to diverging outcomes (the whole point of doing otherwise-matched simulations with some parameter varying), the overall trajectory of the simulation will be begin to exercise the PRNG differently. When *different* events occur between the samples, this does not matter - one random deviate is as good as another. However, diverging trajectories can still share some of the same events. These events should be resolved *consistently*: for example, if a probabilistic threshold is increasing across scenarios, then a particular event testing that should only change from pass to fail. In practice, this means that same events need have the same PRNG draws, which is not possible if the PRNG state has otherwise diverged due to other parts of the simulation.

The HBM PRNG approach encodes events such that when they are the same (as defined by the simulation), they create identical hashes, which are then used to set the PRNG state. This ensure the same subsequent draws for that event.

Examples

```
salt <- 8675309
evt <- list(type = "infection", from = 1, to = 2, time = 3.1)
evt2 <- list(type = "recovery", from = 1, to = 2, time = 3.1)
salt |> hash_seed(evt$type, evt$from, evt$to, evt$time)
print(runif(10))
print(runif(10))
salt |> hash_seed(evt$type, evt$from, evt$to, evt$time)
print(runif(10))
salt <- 42
salt |> hash_seed(evt$type, evt$from, evt$to, evt$time)
print(runif(10))
```

Index