Package ‘ironseed’

July 22, 2025

Title Improved Random Number Generator Seeding

Version 0.1.0

Description A procedure for seeding R's built in random number generators using a variable-length sequence of values. Accumulates input entropy into a 256-bit hash digest or ``ironseed'' and is able to generate a variable-length sequence of output seeds from an ironseed.

License MIT + file LICENSE

Language en-US

Encoding UTF-8

RoxygenNote 7.3.2

Biarch TRUE

NeedsCompilation yes

URL https://github.com/reedacartwright/ironseed

BugReports https://github.com/reedacartwright/ironseed/issues

Suggests tinytest

Author Reed Cartwright [aut, cre] (ORCID: <https://orcid.org/0000-0002-0837-9380>)

Maintainer Reed Cartwright <racartwright@gmail.com>

Repository CRAN

Date/Publication 2025-07-11 08:40:02 UTC

Contents

ironseed *Ironseed: Improved Random Number Generator Seeding*

Description

An ironseed is a 256-bit hash digest constructed from a variable-length input sequence and can be used to generate a variable-length output sequence of seeds, including initializing R's built-in random number generator.

- `ironseed()` creates an ironseed from user supplied objects or automatically from multiple sources of entropy on the local system. It also initializes R's built-in random number generator from an ironseed.
- `create_seqseq()` uses an ironseed to generate a sequence of 32-bit seeds.
- `is_ironseed()` tests whether an object is an ironseed, and `is_ironseed_str()` tests if it is a string representing and ironseed.
- `as_ironseed()` casts an object to an ironseed, and `parse_ironseed_str()` parses a string to an ironseed.

Usage

```
ironseed(..., set_seed = !has_random_seed(), quiet = FALSE)

create_seedseq(fe, n)

is_ironseed(x)

is_ironseed_str(x)

as_ironseed(x)

parse_ironseed_str(x)
```

Arguments

`...`	objects
`set_seed`	a logical indicating whether to initialize `.Random.seed`.
`quiet`	a logical indicating whether to silence messages.
`fe`	an ironseed
`n`	a scalar integer specifying the number of seeds to generate
`x`	a string, ironseed, or other object

Details

Ironseeds have a specific string representation, e.g. "rBQSjhjYv1d-z8dfMATEicf-sw1NSWAvVDi-bQaKSKKQmz1", where each element is a 64-bit number encoded in little-endian base58 format.

Parameter `set_seed` defaults to `TRUE` if `.Random.seed` does not already exist and `FALSE` otherwise.

Ironseed behaves differently depending on the number of arguments passed as `...`.

- 0 arguments: If initialization is enabled, `ironseed()` generates an automatic ironseed. Otherwise, `ironseed()` returns the last ironseed used to initialize `.Random.seed`.
- 1 argument: `ironseed(NULL)` generates an automatic ironseed. For `ironseed(x)`, if `x` is an ironseed object, it is used as is. If `x`, is a scalar character that matches an ironseed string, it is parsed to an ironseed. Otherwise, `x` hashed to create an ironseed.
- 2+ arguments: `ironseed(x,y,...)` hashes the arguments to create an ironseed.

An ironseed is a finite-entropy (or fixed-entropy) hash digest that can be used to generate an unlimited sequence of seeds for initializing the state of a random number generator. It is inspired by the work of M.E. O'Neill and others.

An ironseed is a 256-bit hash digest constructed from a variable-length sequence of 32-bit inputs. Each ironseed consists of eight 32-bit sub-digests. The sub-digests are 32-bit multilinear hashes that accumulate entropy from the input sequence. Each input is included in every sub-digest. The coefficients for the multilinear hashes are generated by a Weyl sequence.

Multilinear hashes are also used to generate an output seed sequence from an ironseed. Each 32-bit output value is generated by uniquely hashing the sub-digests. The coefficients for the output are generated by a second Weyl sequence.

Value

An ironseed. If `.Random.seed` was initialized, the ironseed used will be returned invisibly.

References

- O'Neill (2015) Developing a seed_seq Alternative. https://www.pcg-random.org/posts/developing-a-seed_seq-alternative.html
- O'Neill (2015) Simple Portable C++ Seed Entropy. https://www.pcg-random.org/posts/simple-portable-cpp-seed-entropy.html
- O'Neill (2015) Random-Number Utilities. https://gist.github.com/imneme/540829265469e673d045
- Lemire and Kaser (2018) Strongly universal string hashing is fast. http://arxiv.org/pdf/1202.4961
- Weyl Sequence https://en.wikipedia.org/wiki/Weyl_sequence

See Also

.Random.seed

Examples

```
# Generate an ironseed with user supplied data
ironseed::ironseed("Experiment", 20251031, 1)

# Generate an ironseed automatically and initialize `.Random.seed` with it
ironseed::ironseed(set_seed = TRUE)
```

Index