

# Package ‘SeedMaker’

May 7, 2026

**Title** Generate a Collection of Seeds from a Single Seed

**Version** 1.0.0

**Description** A mechanism for easily generating and organizing a collection of seeds from a single seed, which may be subsequently used to ensure reproducibility in processes/pipelines that utilize multiple random components (e.g., trial simulation).

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Suggests** devtools (>= 2.4.5), knitr (>= 1.45), rmarkdown (>= 2.25), testthat (>= 3.2.0), usethis (>= 2.2.2)

**Imports** checkmate (>= 2.3.0), dplyr (>= 1.1.4), rlang (>= 1.1.2), tibble (>= 3.2.1), tidyr (>= 1.3.0)

**URL** <https://github.com/hdhshowalter/SeedMaker>

**BugReports** <https://github.com/hdhshowalter/SeedMaker/issues>

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Hollins Showalter [aut, cre],  
Eli Lilly and Company [cph, fnd]

**Maintainer** Hollins Showalter <hollins.showalter@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-01-17 09:00:02 UTC

## Contents

|                      |          |
|----------------------|----------|
| seed_gen . . . . .   | 2        |
| seed_maker . . . . . | 3        |
| <b>Index</b>         | <b>8</b> |

---

|          |                                   |
|----------|-----------------------------------|
| seed_gen | <i>Generate a vector of seeds</i> |
|----------|-----------------------------------|

---

### Description

seed\_gen() generates a vector of seeds from a single seed.

### Usage

```
seed_gen(seed, n, sample_vector = 1:1e+06)
```

### Arguments

|               |   |
|---------------|---|
| seed          | A number, interpreted as an integer, or NULL (see <a href="#">set.seed()</a> ). |
| n             | The number of seeds to generate.  |
| sample_vector | The values from which to sample (without replacement).                          |

### Details

length(sample\_vector) must be  $\geq n$ .

The values of sample\_vector must be unique, positive integers.

### Value

An integer vector containing n randomly drawn values from sample\_vector.

### Examples

```
seeds1 <- seed_gen(  
  seed = 1234,  
  n = 10  
)  
  
seeds2 <- seed_gen(  
  seed = 1234,  
  n = 10,  
  sample_vector = 1:100  
)
```

---

|            |  |
|------------|--|
| seed_maker | <i>Construct a list containing a collection of seeds</i> |
|------------|--|

---

### Description

seed\_maker() constructs a hierarchical list containing a collection of seeds from a single seed.

### Usage

```
seed_maker(
  seed,
  level_names,
  n_per_level,
  sample_vector = rep(list(rlang::expr(1:1e+06)), length(level_names)),
  index_level = rep(TRUE, length(level_names)),
  flatten = FALSE
)
```

### Arguments

|               |  |
|---------------|--|
| seed          | A number, interpreted as an integer, or NULL (see <a href="#">set.seed()</a> ).  |
| level_names   | A character vector, or list of character vectors, indicating the element name(s) to use for each level of the returned list.                                     |
| n_per_level   | An integer vector indicating the number of elements to produce at each level of the returned list.   |
| sample_vector | A list of expressions that are evaluable as integer vectors indicating the values from which to sample (without replacement) at each level of the returned list. |
| index_level   | A logical vector indicating whether to index the root name of a respective level.  |
| flatten       | "Flatten" the lowest level of the returned list into a named vector?   |

### Details

length(level\_names) must be equal to length(n\_per\_level) must be equal equal to length(sample\_vector) must be equal to length(index\_level). When these arguments are greater than length 1, the intermediate seeds used to produce the seeds at the lowest level of the returned list are not included therein. In other words, only the collection of seeds at the lowest list level are returned.

If level\_names is a character vector, each of its elements is used as the root name for the list elements produced at the respective level. See seeds\_list1 and seeds\_list2 in **Examples**.

If level\_names is a list of character vectors:

- The element contained in a length 1 vector is used as the root name for the list elements produced at the respective level.
- Elements contained in a vector that is greater than length 1 are used verbatim as the names for the list elements produced at the respective level. The corresponding value of n\_per\_level must match the length of the vector.

See `seeds_list3`, `seeds_list4`, and `seeds_list5` in **Examples**.

For each element, `i`, of `sample_vector`:

- `length(eval(sample_vector[[i]]))` must be  $\geq n\_per\_level[i]$ .
- The values of `eval(sample_vector[[i]])` must be unique, positive integers.

See `seeds_list6`, `seeds_list7`, and `seeds_list8` in **Examples**.

Element name indexing applies to root name-provided levels only. Thus, for each element, `i`, of `index_level`, if `length(level_names[[i]])` is greater than 1, the value of `index_level[i]` is ignored. Otherwise:

- When `index_level[i]` is `TRUE`, the element names at the respective level are indexed from  $1 - n\_per\_level[i]$ .
- When `index_level[i]` is `FALSE`, the element name at the respective level is used verbatim, so long as `n_per_level[i] = 1`, since suppressing element name indexing makes sense only for single-element levels.

See `seeds_list9`, `seeds_list10`, `seeds_list11`, `seeds_list12`, and `seeds_list13` in **Examples**.

When `flatten` is set to `TRUE`, each set of seeds at the lowest level of the returned list is converted into a named vector (versus remaining in the default structure, which is a list of single value elements). When only a single level is specified (via `level_names`, `n_per_level`, `sample_vector`, and `index_level` all being length 1), a named integer vector is returned instead of a list. See `seeds_flat1`, `seeds_flat2`, `seeds_flat5`, `seeds_flat9`, `seeds_flat10`, and `seeds_flat11` in **Examples**.

## Value

When `flatten` is set to `FALSE`, a hierarchical list with `length(level_names)` levels containing a collection of seeds.

When `flatten` is set to `TRUE`, a hierarchical list with `length(level_names) - 1` level(s) containing a collection of seeds (where "0 levels" corresponds to a named integer vector - see **Details**).

## Examples

```
seeds_list1 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(10)
)

seeds_list2 <- seed_maker(
  seed = 1234,
  level_names = c("sim", "dataset"),
  n_per_level = c(10, 15)
)

seeds_list3 <- seed_maker(
  seed = 1234,
  level_names = list("sim", c("component_a", "component_b", "component_c")),
```

```
    n_per_level = c(10, 3)
  )

try(seeds_list4 <- seed_maker(
  seed = 1234,
  level_names = list("sim", c("component_a", "component_b", "component_c")),
  n_per_level = c(10, 4)
))

seeds_list5 <- seed_maker(
  seed = 1234,
  level_names = list(
    "sim",
    c("component_a", "component_b", "component_c"),
    "dataset"
  ),
  n_per_level = c(10, 3, 5)
)

seeds_list6 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(10),
  sample_vector = list(1:100)
)

seeds_list7 <- seed_maker(
  seed = 1234,
  level_names = c("sim", "dataset"),
  n_per_level = c(10, 15),
  sample_vector = list(1:100, 1:1000)
)

seeds_list8 <- seed_maker(
  seed = 1234,
  level_names = list(
    "sim",
    c("component_a", "component_b", "component_c"),
    "dataset"
  ),
  n_per_level = c(10, 3, 5),
  sample_vector = list(1:100, 1:1000, 1:10000)
)

seeds_list9 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(1)
)

try(seeds_list10 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
```

```
n_per_level = c(10),
index_level = FALSE
))

seeds_list11 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(1),
  index_level = FALSE
)

seeds_list12 <- seed_maker(
  seed = 1234,
  level_names = list("sim", c("component_a", "component_b", "component_c")),
  n_per_level = c(1, 3),
  index_level = c(FALSE, TRUE)
)

seeds_list13 <- seed_maker(
  seed = 1234,
  level_names = list("sim", c("component_a", "component_b", "component_c")),
  n_per_level = c(1, 3),
  index_level = c(FALSE, FALSE)
)

identical(seeds_list12, seeds_list13)

seeds_flat1 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(10),
  flatten = TRUE
)

is.list(seeds_flat1)
is.integer(seeds_flat1)

seeds_flat2 <- seed_maker(
  seed = 1234,
  level_names = c("sim", "dataset"),
  n_per_level = c(10, 15),
  flatten = TRUE
)

is.list(seeds_flat2$sim1)
is.integer(seeds_flat2$sim1)

seeds_flat5 <- seed_maker(
  seed = 1234,
  level_names = list(
    "sim",
    c("component_a", "component_b", "component_c"),
    "dataset"
  )
)
```

```
    ),
    n_per_level = c(10, 3, 5),
    flatten = TRUE
  )

is.list(seeds_flat5$sim1$component_a)
is.integer(seeds_flat5$sim1$component_a)

seeds_flat9 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(1),
  flatten = TRUE
)

is.list(seeds_flat9)
is.integer(seeds_flat9)

try(seeds_flat10 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(10),
  index_level = FALSE,
  flatten = TRUE
))

seeds_flat11 <- seed_maker(
  seed = 1234,
  level_names = c("sim"),
  n_per_level = c(1),
  index_level = FALSE,
  flatten = TRUE
)

is.list(seeds_flat11)
is.integer(seeds_flat11)
```

# Index

seed\_gen, [2](#)  
seed\_maker, [3](#)  
set.seed(), [2](#), [3](#)