Package: gsample (via r-universe)

October 13, 2024

Type Package

Title Efficient Weighted Sampling Without Replacement

Version 0.1.0

Author Valerio Gherardi

Maintainer Valerio Gherardi <vgherard@sissa.it>

Description Sample without replacement using the Gumbel-Max trick
(c.f. \url{https://arxiv.org/pdf/1903.06059.pdf}).

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

LinkingTo Rcpp

Imports Rcpp

SystemRequirements C++11

URL https://github.com/vgherard/gsample

BugReports https://github.com/vgherard/gsample/issues

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

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

RemoteUrl https://github.com/vgherard/gsample

RemoteRef HEAD

RemoteSha 5b1bbc8e768417389b6ca1451e80f93457cbcfba

Contents

gsample *Efficient weighted sampling without replacement*

Description

These functions offer a drop-in replacement for sample, with considerably better performance for the case of weighted sampling without replacement (both from the speed and memory point of view). The interface of `gsample` and `gsample.int` is essentially the same of the corresponding `base` functions, so that they can be replaced in base R code with little (if any) modification.

Usage

```
gsample.int(n, size = n, replace = FALSE, prob = NULL, algorithm = NULL)

gsample(x, size, replace = FALSE, prob)
```

Arguments

`n`	length one integer. The total number of categories to choose from. See ‘Details’.
`size`	length one integer. Size of sample.
`replace`	TRUE or FALSE. Sample with replacement? Defaults to `FALSE`.
`prob`	either `NULL`, or a numeric vector of length `n`, containing probability weights for sampling the various classes. If `NULL` (default), sampling is performed assuming uniform probabilities.
`algorithm`	either `NULL`, `"introselect"` or `"partial_heap"`. The default (`NULL`), uses a rough estimate of the relative time complexities to select between the two algorithms. See ‘Details’.
`x`	a vector of one or more elements from which to choose.

Details

These functions are meant to replace `base::sample()` and `base::sample.int()` for weighted sampling without replacement, for which the `base` implementation is inefficient. For uniform sampling, or sampling with replacement, `gsample` simply calls the `base` functions.

The APIs of `gsample()` and `gsample.int()` are almost identical to the ones of `base::sample()` and `base::sample.int()`, respectively, with the following differences:

- The argument `useHash` for `base::sample.int(replace = FALSE, prob = NULL)` is not provided.
- An additional `algorithm` argument.

The basic arguments `x`, `n`, `size`, `replace` and `prob` are documented in sample, to which we refer. Here we describe the additional argument `algorithm`.

`gsample` supports two algorithms, `"introselect"` and `"partial_heap"` with different space and time complexities for weighted sampling without replacement. If the argument `algorithm` is left as default (NULL), gsample() tentatively selects the fastest algorithm based on a rough estimate of the two running times. algorithm = "introselect" has time complexity *T = O(n)*, and space complexity *S = O(n)*, whereas algorithm = "partial_heap" has time complexity *T = O(n * log(size))*, and space complexity *S = O(size)*. The running time of "introselect" is largely independent of the actual value of prob, whereas "partial_heap" can be get some speed-up (speed-down) if prob is sorted in decreasing (increasing) order.

Despite its worst asymptotic performance, "partial_heap" is typically faster for small sample sizes (of less than 100, say), in which case it is also much cheaper from the memory point of view.

Value

an integer vector of class indexes for gsample.int, a vector of the same type of x for gsample.

Author(s)

Valerio Gherardi

Examples

```
set.seed(840)
gsample(letters, 5, prob = runif(length(letters)))
gsample.int(10, 3, prob = 10:1)
```

Index