Randomisation of Rows in a Data Frame

Smit, A. J.
University of the Western Cape

2025-07-11

Table of contents

Randomising the order of observations is effectively a permutation of the data.frame's (row) index vector. Because R stores a data.frame as a list of equal-length vectors, shuffling the rows amounts to reordering each vector simultaneously according to a single, randomly drawn permutation.

1 Base-R

```r
set.seed(42)
df_perm <- df[sample(nrow(df)), ]
row.names(df_perm) <- NULL
```

Line 1

Fix the RNG for reproducibility when needed

Line 2

`sample()` without the `replace` argument returns a permutation

Line 3

(Optional) drop the original row indices

`sample(nrow(df))` results in a vector `1:n` rearranged into a fresh random order, so subsetting the data.frame with that vector coerces every column to follow suit. The row names stay attached unless one removes them.

2 Tidyverse

```r
library(dplyr)

df_perm <- df %>%
  slice_sample(n = nrow(.))
```

```r
# or, predating slice_sample():
df_perm <- df %>%
  sample_frac(1)
```

`slice_sample()` accepts an optional `weight_by` argument, permitting unequal permutation weights should the null hypothesis require them. By contrast, `sample_frac(1)` asks to "take a 100 % sample", and, being fraction-based, it sidesteps the need to compute `nrow(df)`.

3 Statistical context

Within permutation-based inference, randomising rows removes any systematic linkage between predictor and response variables but it conserves the empirical marginal distributions. Recomputing a test statistic over many such reshufflings results in distributions that approximate the data's sampling distribution under the null hypothesis of no association. One call to `sample()` is sufficient to create one permutation. By placing that call inside `replicate()` or `purrr::rerun()` supplies the Monte-Carlo ensemble.

Randomisation of rows presupposes exchangeability. If one's data carry temporal, spatial, or hierarchical strata, a naïve global shuffle may void the null model's statistical validity. In such cases one would confine `sample()` to blocks delineated by the relevant grouping variable, for example via `dplyr::group_modify(~ .x[sample(nrow(.x)), ])`, and so restrict permutations to contextually relevant partitions.

Bibliography