

Package: prspace (via r-universe)

September 27, 2026

Title Probability Spaces with the Same Syntax in R and Python

Version 0.2.0

Description Write probability statements such as 'Pr[X == 0] <- 0.25' and read them back with 'Pr[X == 0]'. Events can declare their support (real, integer, count, or a finite set), and anything the rules of probability then pin down, such as complements and totals, is derived. Assignments that contradict earlier ones are flagged with the range of values that would have been consistent. A companion Python package on PyPI offers the same syntax.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL <https://github.com/ivanharvard/prspace>

BugReports <https://github.com/ivanharvard/prspace/issues>

Repository <https://ivanharvard.r-universe.dev>

Date/Publication 2026-09-27 00:13:52 UTC

RemoteUrl <https://github.com/ivanharvard/prspace>

RemoteRef HEAD

RemoteSha 6fc1a5a14f95355b95eb8f722ae0bd3b0f9da6c5

RemoteSubdir r

Contents

Event	2
PrSpace	3
Index	5

Event	<i>Create an event</i>
-------	------------------------

Description

An Event is a quantity you make probability statements about. Comparing it with a single value ($=$, $!=$, $<$, $<=$, $>$, $>=$) gives a condition such as $S_B == 0$, which is what you index a [PrSpace](#) with. $!(X == 0)$ is the complement $X != 0$.

The support says which values the event can take, which lets a PrSpace derive and check more. For example, with `support = "count"`, $S_B >= 1$ is the complement of $S_B == 0$ and $\text{Pr}[S_B < 0]$ is 0.

If name is omitted, it is taken from the variable the event is bound to the first time it is compared, so `S_B <- Event()` prints as S_B.

Usage

```
Event(name = NULL, support = NULL)
```

Arguments

name	Optional label used when printing, a single string.
support	"real" (the default when NULL), "integer", "count" (0, 1, 2, ...), or a vector of the values the event can take, such as <code>c(1, 2, 3)</code> or <code>c("H", "T")</code> . Events with string values can only be compared with $=$ and $!=$.

Value

A `prspace_event` object.

See Also

[PrSpace](#)

Examples

```
X <- Event()
X == 0
!(X < 3)
Event("S_B", support = "count")
Event("Coin", support = c("H", "T"))
```

PrSpace

*Create a probability space***Description**

A PrSpace stores probabilities of conditions built from [Events](#). Assign with `Pr[X == 0] <- p` (or `=`) and read back with `Pr[X == 0]`. Remove a value with `Pr[X == 0] <- NULL`.

Reading a condition returns, in order: the value assigned to it or to any condition describing the same event; one minus the value assigned to its complement; 0 or 1 if the event's support makes it impossible or certain; and, with `derive = TRUE`, any value the rules of probability pin down given everything assigned to that event. Otherwise it signals an error of class `prspace_undefined` whose fields `lower` and `upper` give the range the value is known to lie in.

Assigning a condition replaces any stored value for the same event or its complement. With `check on`, the new value is compared with the range consistent with the other stored values, and a violation signals a condition of class `prspace_inconsistent`.

A PrSpace has reference semantics: modifying it inside a function modifies the caller's copy.

Usage

```
PrSpace(check = "warn", derive = TRUE)
```

Arguments

<code>check</code>	What to do when an assignment is inconsistent with the others: "warn" (store it and warn), "error" (refuse it), or "off".
<code>derive</code>	If TRUE, derive any value the assigned ones determine. If FALSE, only assigned values, their complements, and impossible or certain events are returned.

Value

A `prspace_space` object.

See Also

[Event](#)

Examples

```
Pr <- PrSpace()
S_B <- Event(support = "count")
n_samples <- choose(19, 10)
for (B in c(100, 1000)) {
  Pr[S_B == 0] <- prod(1 - (0:(B - 1) / n_samples))
  cat("B =", B, " Pr[S_B >= 1] =", Pr[S_B >= 1], "\n")
}
Pr
```

```
X <- Event(support = c(1, 2, 3))
Pr[X == 1] <- 0.2
Pr[X == 2] <- 0.3
Pr[X == 3]

Y <- Event()
Pr[Y < 3] <- 0.6
tryCatch(Pr[Y < 5] <- 0.4, warning = function(w) conditionMessage(w))
```

Index

Event, [2](#), [3](#)

PrSpace, [2](#), [3](#)