

# Package ‘broadcast’

January 5, 2025

**Title** Simple Broadcasted Binary Operations for Atomic Arrays with Minimal Dependencies

**Version** 0.0.0.9000

**Description** Implements simple broadcasted operations for atomic arrays.

Includes relational operators (``==``, ``!=``, ``<``, ``>``, ``<=``, ``>=``),  
boolean combiner operators (``&``, ``|``, `xor()`, ``not-and``),  
integer/double/complex arithmetic operators (``+``, ``-``, ``*``, ``/``, ``^``, ``%%``, `pmin()`, `pmax()`),  
and string concatenation operators (``+``, ``*``).

The broadcasted operators have about the same performance as their non-broadcasted base 'R' counterparts.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LinkingTo** Rcpp

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Depends** R (>= 4.2.0)

**Imports** Rcpp (>= 1.0.11)

**Suggests** tinytest

## Contents

aaa00_broadcast_help . . . . .	1
array_recycle . . . . .	2
atomic_typecast . . . . .	3
bc . . . . .	4
bc.d . . . . .	5
bc_pred_dim . . . . .	6

<b>Index</b>	7
--------------	---

---

aaa00_broadcast_help	<i>broadcast: Subset Methods as Alternatives to the Square Brackets Operators for Programming</i>
----------------------	---

---

### Description

broadcast:

Simple broadcasted infix operations for atomic arrays in 'R'.

The 'broadcast' package supports relational operators (==, !=, <, >, <=, >=), logical combiners (&, |, xor, nand), arithmetic (+, -, \*, /, ^)

### Author(s)

**Author, Maintainer:** Tony Wilkes <tony\_a\_wilkes@outlook.com> ([ORCID](#))

### References

The badges shown in the documentation of this R-package were made using the services of: <https://shields.io/>

---

array_recycle	<i>Recycle Array Dimensions</i>
---------------	---------------------------------

---

### Description

The array\_recycle() function recycles array dimensions until the specified dimension sizes are reached, and returns the array.

The various broadcasting functions "recycle" an array virtually, meaning little to no additional memory is needed.

The array\_recycle() function, however, physically recycles an array (and thus actually occupies memory space).

### Usage

```
array_recycle(x, tdim)
```

### Arguments

x	an atomic or recursive array or matrix.
tdim	an integer vector, giving the target dimension to reach.

### Value

Returns the recycled array.

## Examples

```
x <- matrix(1:9, 3,3)
colnames(x) <- LETTERS[1:3]
rownames(x) <- letters[1:3]
names(x) <- month.abb[1:9]
print(x)

array_recycle(x, c(3,3,2)) # recycle to larger size
```

---

atomic\_typecast

*Atomic Type Casting With Names and Dimensions Preserved*

---

## Description

Atomic type casting in R is generally performed using the functions [as.logical](#), [as.integer](#), [as.double](#), [as.character](#), [as.complex](#), and [as.raw](#).

Converting an object between atomic types using these functions strips the object of its attributes, including (dim)names and dimensions.

The functions provided here by the 'tinycodet' package preserve the dimensions, dimnames, and names.

The functions are as follows:

- `as_bool()`: converts object to atomic type logical (TRUE, FALSE, NA).
- `as_int()`: converts object to atomic type integer.
- `as_dbl()`: converts object to atomic type double (AKA decimal numbers).
- `as_chr()`: converts object to atomic type character.
- `as_cplx()`: converts object to atomic type complex.
- `as_raw()`: converts object to atomic type raw.

## Usage

```
as_bool(x, ...)

as_int(x, ...)

as_dbl(x, ...)

as_chr(x, ...)

as_cplx(x, ...)

as_raw(x, ...)
```

## Arguments

x	vector, matrix, array (or a similar object where all elements share the same type).
...	further arguments passed to or from other methods.

**Value**

The converted object.

**Examples**

```
# matrix example ====
x <- matrix(sample(-1:28), ncol = 5)
colnames(x) <- month.name[1:5]
rownames(x) <- month.abb[1:6]
names(x) <- c(letters[1:20], LETTERS[1:10])
print(x)

as_bool(x)
as_int(x)
as_dbl(x)
as_chr(x)
as_cplx(x)
as_raw(x)

#####

# factor example ====
x <- factor(month.abb, levels = month.abb)
names(x) <- month.name
print(x)

as_bool(as_int(x) > 6)
as_int(x)
as_dbl(x)
as_chr(x)
as_cplx(x)
as_raw(x)
```

---

bc

*Broadcasted Relational Operations on Atomic Arrays*


---

**Description**

The `bc()` function performs broadcasted relational operations on 2 atomic arrays.

**Usage**

```
bc(x, y, op, prec = sqrt(.Machine$double.eps))
```

**Arguments**

<code>x, y</code>	conformable atomic arrays of types.
<code>op</code>	a single string, giving the operator. Supported operators: <code>==</code> , <code>!=</code> , <code>&lt;</code> , <code>&gt;</code> , <code>&lt;=</code> , <code>&gt;=</code> .

`prec` a single decimal number, giving the machine precision.  
Only applicable when `x` and/or `y` is of type double

### Value

A numeric array as a result of the broadcasted arithmetic operation.

### Examples

```
x.dim <- c(10:8)
x.len <- prod(x.dim)
x.data <- sample(c(NA, 1.1:1000.1), x.len, TRUE)
x <- array(x.data, x.dim)
y <- array(1:50, c(10,1,1))

bc(x, y, "==")
bc(x, y, "!=")
bc(x, y, "<")
bc(x, y, ">")
bc(x, y, "<=")
bc(x, y, ">=")
```

---

bc.d

*Broadcasted Decimal Arithmetic Operations on Atomic Arrays*


---

### Description

The `bc.d()` function performs broadcasted decimal arithmetic operations on 2 atomic arrays.

### Usage

```
bc.d(x, y, op)
```

### Arguments

`x, y` conformable atomic arrays of types logical, integer, or double.  
`op` a single string, giving the operator.  
Supported operators: `+`, `-`, `*`, `/`, `^`, `pmin`, `pmax`.

### Value

A numeric array as a result of the broadcasted arithmetic operation.

**Examples**

```

x.dim <- c(10:8)
x.len <- prod(x.dim)
x.data <- sample(c(NA, 1.1:1000.1), x.len, TRUE)
x <- array(x.data, x.dim)
y <- array(1:50, c(10,1,1))

bc.d(x, y, "+")
bc.d(x, y, "-")
bc.d(x, y, "*")
bc.d(x, y, "/")
bc.d(x, y, "^")

```

---

bc_pred_dim	<i>Predict Broadcasted dimensions</i>
-------------	---------------------------------------

---

**Description**

```
bc_pred_dim()
```

**Usage**

```
bc_pred_dim(x, y)
```

**Arguments**

`x, y`                      an atomic array or matrix.

**Value**

Returns the recycled array.

**Examples**

```

x <- matrix(1:9, 3,3)
colnames(x) <- LETTERS[1:3]
rownames(x) <- letters[1:3]
names(x) <- month.abb[1:9]
print(x)

array_recycle(x, c(3,3,2)) # recycle to larger size

```

# Index

[aaa00\\_broadcast\\_help](#), 1  
[array\\_recycle](#), 2  
[as.character](#), 3  
[as.complex](#), 3  
[as.double](#), 3  
[as.integer](#), 3  
[as.logical](#), 3  
[as.raw](#), 3  
[as\\_bool \(atomic\\_typecast\)](#), 3  
[as\\_chr \(atomic\\_typecast\)](#), 3  
[as\\_cplx \(atomic\\_typecast\)](#), 3  
[as\\_dbl \(atomic\\_typecast\)](#), 3  
[as\\_int \(atomic\\_typecast\)](#), 3  
[as\\_raw \(atomic\\_typecast\)](#), 3  
[atomic\\_typecast](#), 3  
  
[bc](#), 4  
[bc.d](#), 5  
[bc\\_pred\\_dim](#), 6  
[broadcast \(aaa00\\_broadcast\\_help\)](#), 1  
[broadcast-package](#)  
    ([aaa00\\_broadcast\\_help](#)), 1  
[broadcast\\_help \(aaa00\\_broadcast\\_help\)](#), 1