

# Package ‘binaryLogic’

December 13, 2017

**Title** Binary Logic

**Description** Provides the binary S3 class. The instance of binary is used to convert a decimal number (Base10) to a binary number (Base2). The Class provides some features e.G. shift(), rotate(), summary(). Based on logical vectors.

**Version** 0.3.9

**Maintainer** Daniel Dörrhöfer <ddo@openmailbox.org>

**URL** <https://github.com/d4ndo/binaryLogic>

**BugReports** <https://github.com/d4ndo/binaryLogic/issues>

**Depends** R (>= 3.0.2)

**Suggests** testthat

**License** GPL-3

**LazyData** true

**Encoding** UTF-8

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Daniel Dörrhöfer [aut, cre],  
Aaron Rudkin [ctb]

**Repository** CRAN

**Date/Publication** 2017-12-13 10:36:49 UTC

## R topics documented:

as.binary . . . . .	2
bin2gray . . . . .	3
binAdd . . . . .	4
binary . . . . .	4
binaryLogic . . . . .	6
binaryPrefix . . . . .	7
binSeq . . . . .	7
byte . . . . .	8

bytesNeeded . . . . .	9
fillUpToBit . . . . .	9
fillUpToByte . . . . .	10
gray2bin . . . . .	11
is.binary . . . . .	11
loadAttributes . . . . .	12
negate . . . . .	12
Ops.binary . . . . .	13
print.binary . . . . .	13
rotate . . . . .	14
saveAttributes . . . . .	14
shiftLeft . . . . .	15
shiftRight . . . . .	15
summary.binary . . . . .	16
switchEndianess . . . . .	17

## Index 18

---

as.binary	<i>as binary digit.</i>
-----------	-------------------------

---

### Description

Converts an integer (Base10) to a binary (Base2) number. It also converts a logical vector to a binary (Base2) number (see examples).

### Usage

```
as.binary(x, signed=FALSE, littleEndian=FALSE, size=2, n=0, logic=FALSE)
```

### Arguments

x	integer or logical vector.
signed	TRUE or FALSE. Unsigned by default. (two's complement)
littleEndian	if TRUE. Big Endian if FALSE.
size	in Byte. Needed if <b>signed</b> is set. (by default 2 Byte)
n	in Bit. Can be set if <b>unsigned</b> is set to TRUE. (by default 0 Bit = auto)
logic	If set to TRUE, x is expected as logical vector.

### Details

The binary number is represented by a logical vector. The bit order usually follows the same endianness as the byte order. No floating-point support. If logic is set to TRUE an integer vector is interpreted as a logical vector (>0 becomes TRUE and 0 becomes FALSE)

- Little Endian (LSB) —> (MSB)
- Big Endian (MSB) <— (LSB)

Auto switch to signed if num < 0.

**Value**

a vector of class binary.

**See Also**

[is.binary](#) and [binary](#)

**Examples**

```
as.binary(0xAF)
as.binary(42)
as.binary(42, littleEndian=TRUE)
as.binary(c(0xAF, 0xBF, 0xFF))
as.binary(c(2,4,8,16,32), signed=TRUE, size=1)
as.binary(-1, signed=TRUE, size=1)
as.binary(1:7, n=3)
as.binary(sample(2^8,3),n=8)
as.binary(c(1,1,0), signed=TRUE, logic=TRUE)
as.binary(c(TRUE,TRUE,FALSE), logic=TRUE)
```

---

bin2gray

*A gray code converter function*

---

**Description**

This function converts a binary number (base2) to a gray code

**Usage**

```
bin2gray(x)
```

**Arguments**

x                    The binary number (base2) or a logical vector.

**Value**

The gray code as logical vector.

**See Also**

[gray2bin](#)

binAdd

*Binary Addition (+)*

---

**Description**

Adds two binary numbers. (x + y)

**Usage**

```
binAdd(x, y)
```

**Arguments**

x	summand 1 (binary vector)
y	summand 2 (binary vector)

**Details**

Little-Endian and unsigned is not supported at the moment. No floating point supported. if x or y is signed the return value will also be signed.

**Value**

The sum of x and y. Returns a binary vector.

**See Also**

base::as.logical , base::is.logical, base::raw

**Examples**

```
five <- as.binary(5); ten <- as.binary(10);  
as.numeric(binAdd(ten, five))  
binAdd(as.binary(c(0,1), logic=TRUE), as.binary(c(1,0), logic=TRUE))
```

---

binary

*Binary digit.*

---

**Description**

Create objects of type binary.

**Usage**

```
binary(n, signed=FALSE, littleEndian=FALSE)
```

**Arguments**

n	length of vector. Number of bits
signed	TRUE or FALSE. Unsigned by default. (two's complement)
littleEndian	if TRUE. Big Endian if FALSE.

**Details**

The binary number is represented by a *logical* vector. The bit order usually follows the same endianness as the byte order. How to read:

- Little Endian (LSB)  $\rightarrow$  (MSB)
- Big Endian (MSB)  $\leftarrow$  (LSB)

The Big Endian endianness stores its MSB at the lowest address. The Little Endian endianness stores its MSB at the highest address.

e.g. `b <- binary(8)`.

- "Little Endian" : MSB at `b[1]` and LSB at `b[8]`.
- "Big Endian" : LSB at `b[1]` and MSB at `b[8]`.

No floating-point support.

**Value**

a vector of class `binary` of length `n`. By default filled with `zeros(0)`.

**See Also**

[as.binary](#) and [is.binary](#).

**Examples**

```
b <- binary(8)
summary(b)
b <- binary(16, signed=TRUE)
summary(b)
b <- binary(32, littleEndian=TRUE)
summary(b)
```

## Description

This package contains the **binary** S3 class. A data object can be instantiated to store a binary number(Base2).

It can be used to convert, negate, shift or rotate the binary number. (switchEndianess, bytesNeeded, binaryPrefix, fillUpToByte).

Binary operators:

- == , != , < , <= , > , >=
- + , - , ^ , \*
- & , | , xor (Logical Operator. Bitwise operation. The smaller vector is added up with zeros)
- ! (Indicates logical negation (NOT). Bitwise Operations)

binaryLogic functions:

- shiftLeft(binary) , shiftRight(binary)
- rotate(binary)
- negate(binary)
- switchEndianess(binary)

Additional function:

- fillUpToByte, fillUpToBit
- bytesNeeded
- binaryPrefix
- byte

## Details

This **binary** class is just not that great at heavy number crunching, but it brings some benefits. Especially if you like to work using vectors in R. It inherits from the *logical* class. Some function from package **binaryLogic** can be applied to *logical* vectors. Such as shift or rotate (see help).

The internal structure looks like this:

```
structure(c(TRUE, FALSE), class = c("binary", "logical"), signed = FALSE, littleEndian = FALSE)
```

It is composed of a *logical* vector and several attributes. This structure shows a big endian number, it corresponds to the value = 2 (Base10).

---

binaryPrefix	<i>Binary prefix (KiB,MiB,..)</i>
--------------	-----------------------------------

---

**Description**

Num of byte needed to fit in  $n * \text{KiB}$ ,  $\text{MiB}$  ..etc.

**Usage**

```
binaryPrefix(n, prefix="KiB")
```

**Arguments**

n	numeric value
prefix	binary prefix * byte. Expeting a »string«

**Details**

KiB <- KibiByte MiB <- MebiByte GiB <- GibiByte TiB <- TebiByte PiB <- PebiByte EiB <- ExiByte ZiB <- ZebiByte YiB <- YobiByte

**Value**

The number of byte fitting in  $n * \text{binary prefix} * \text{byte}$

**See Also**

[bytesNeeded](#) or [fillUpToByte](#) or [byte](#)

**Examples**

```
#Get the number of byte needed to hold 0.5 and 1:10 KiB
binaryPrefix(c(0.5,1:10),"KiB")
#Get the number of bit needed to hold 1 KiB
binaryPrefix(1,"KiB")*byte()
```

---

binSeq	<i>Binary sequence</i>
--------	------------------------

---

**Description**

Binary sequence.

**Usage**

```
binSeq(x, ...)
```

8

byte

### Arguments

x                    a sequence.  
...                  used for dec2bin().

### Value

a sequence list of binary digits.

### See Also

[binary](#)

### Examples

```
binSeq(0:4)
```

---

byte

*A simple helper function to return the size of one byte*

---

### Description

Used to increase readability

### Usage

```
byte()
```

### Value

The size of one byte (8)

### See Also

[bytesNeeded](#) or [fillUpToByte](#) or [binaryPrefix](#)



---

bytesNeeded	<i>Minimum number of "byte" needed to hold n "bit"</i>
-------------	--

---

**Description**

A simple helper function that returns the minimum number of byte needed to hold the amount of n bit.

**Usage**

```
bytesNeeded(n)
```

**Arguments**

n	The number of bit.
---	--------------------

**Value**

The number of minimum byte needed to hold n bit.

**See Also**

[fillUpToByte](#) or [binaryPrefix](#) or [byte](#)

**Examples**

```
ten <- as.binary(10)
bytesNeeded(length(ten))
```

---

fillUpToBit	<i>Fill up to bit (000..)</i>
-------------	-------------------------------

---

**Description**

Fills up the binary number with zeros(0) or ones(1), to the size n in bit.

**Usage**

```
fillUpToBit(x, n, value=FALSE)
```

**Arguments**

x	The binary number to fill up with zeros. (Any binary vector).
n	size in bit.
value	to fill up with FALSE(0) or fill up with TRUE(1).

**Details**

No floating point supported.

**Value**

binary number. A binary vector with the desired size.

**See Also**

[fillUpToByte](#).

**Examples**

```
fillUpToBit(as.binary(c(1,1), logic=TRUE), n=4)
fillUpToBit(as.binary(c(1,0,1), logic=TRUE), n=4, value=FALSE)
```

---

fillUpToByte	<i>Fill up to Byte (00000000..)</i>
--------------	-------------------------------------

---

**Description**

Fills up the binary number with zeros(0) or ones(1), to the size in Byte.

**Usage**

```
fillUpToByte(x, size=0, value=FALSE)
```

**Arguments**

x	The binary number to fill up with zeros. (Any binary vector).
size	in Byte. 0 = auto (smallest possible Byte).
value	to fill up with FALSE(0) or fill up with TRUE(1).

**Details**

No floating point supported.

**Value**

binary number. A binary vector with the desired size.

**See Also**

[fillUpToBit](#) or [bytesNeeded](#), [negate](#), [switchEndianess](#).

**Examples**

```
fillUpToByte(as.binary(c(1,1), logic=TRUE), size=2)
fillUpToByte(as.binary(c(1,0,1), logic=TRUE), size=2, value=FALSE)
```

---

gray2bin	<i>A gray code to binary converter function</i>
----------	---

---

**Description**

This function converts a gray code to a binary number (base2)

**Usage**

```
gray2bin(x, ...)
```

**Arguments**

x	The gray code as logical vector.
...	Additional parameter for binary()

**Value**

The binary number (base2).

**See Also**

[bin2gray](#)

---

is.binary	<i>is Binary Vector</i>
-----------	-------------------------

---

**Description**

test for object "binary".

**Usage**

```
is.binary(x)
```

**Arguments**

x	object to test.
---	-----------------

**Value**

TRUE or FALSE.

**See Also**

[as.binary](#) and [binary](#)

---

loadAttributes	<i>loadAttributes Helper function load Attributes</i>
----------------	---

---

**Description**

loadAttributes  
 Helper function load Attributes

**Usage**

loadAttributes(x, l)

**Arguments**

x	x
l	l

---

negate	<i>Binary Negation (!)</i>
--------	----------------------------

---

**Description**

Negates the binary number x. Negation x -> -x or -x -> x

**Usage**

negate(x)

**Arguments**

x	The number to be negated. A binary vector is expected.
---	--

**Details**

An »unsigned« number will be returned as »signed« regardless of whether the value is negative. No floating point supported.

**Value**

The negated number of x. Returns a binary vector with signed=TRUE

**See Also**

[switchEndianness](#) or [fillUpToByte](#).

**Examples**

```
summary(negate(as.binary(5, signed=TRUE)))
summary(negate(as.binary(-5, signed=TRUE)))
summary(negate(as.binary(5, signed=FALSE)))
```

---

Ops.binary

*Group Generic Ops*


---

**Description**

Group generic Ops operators

**Usage**

```
## S3 method for class 'binary'
Ops(e1, e2)
```

**Arguments**

e1	e1
e2	e2

---

print.binary

*Print method for binary number.*


---

**Description**

This method prints the binary number.

**Usage**

```
## S3 method for class 'binary'
print(x, ...)
```

**Arguments**

x	any binary number.
...	further arguments.

**Value**

Output in ones and zeros (binary vector).

**See Also**

[summary.binary](#) provides some additional information.

---

rotate	<i>Rotate no carry ()</i>
--------	---------------------------

---

**Description**

A circular shift

**Usage**

```
rotate(x, n)
```

**Arguments**

x	The binary number to rotate. (binary or logical vector).
n	The number of bits to rotate.

**Value**

rotates the vector from left to right. The value from MSB is used to fill up the vector at LSB. Returns a binary/logical vector.

**See Also**

[shiftLeft](#) and [shiftRight](#)

**Examples**

```
x <- as.binary(c(1,0,0,1,1,1,0,1), logic=TRUE); x
rotate(x,1)
rotate(x,2)
```

---

saveAttributes	<i>saveAttributes Helper function save Attributes</i>
----------------	---

---

**Description**

saveAttributes

Helper function save Attributes

**Usage**

```
saveAttributes(x)
```

**Arguments**

x	x
---	---

---

shiftLeft	<i>Binary Left Shift («)</i>
-----------	------------------------------

---

**Description**

Logical left shift  $x \ll n$

**Usage**

```
shiftLeft(x, n)
```

**Arguments**

x	The binary number to shift. (binary or logical vector).
n	The number of bits to shift.

**Value**

Pushes 0's(FALSE) to the vector from right(LSB) to left(MSB). Everything on right(MSB) side drops out. Returns a binary/logical vector

**See Also**

[shiftRight](#) and [rotate](#)

**Examples**

```
x <- as.binary(c(1,0,0,1,1,1,0,1), logic=TRUE); x
shiftLeft(x,1)
shiftLeft(x,2)
```

---

shiftRight	<i>Binary Right Shift (»)</i>
------------	-------------------------------

---

**Description**

Logical right shift  $x \gg n$

**Usage**

```
shiftRight(x, n)
```

**Arguments**

x	The binary number to shift. (binary or logical vector).
n	The number of bits to shift.

**Value**

Pushes 0's(FALSE) to the vector from left(MSB) to right(LSB). Everything on right(LSB) side drops out. Returns a binary/logical vector

**See Also**

[shiftLeft](#) and [rotate](#)

**Examples**

```
x <- as.binary(c(1,0,0,1,1,1,0,1), logic=TRUE); x
shiftRight(x,1)
shiftRight(x,2)
```

---

summary.binary

*Summary method for binary number.*

---

**Description**

This method provides information about the attributes of the binary number.

**Usage**

```
## S3 method for class 'binary'
summary(object, ...)
```

**Arguments**

object	binary number.
...	further arguments.

**Value**

Contains the following information:

- Signedness : unsigned or signed
- Endianess : Big-Endian or Little-Endian
- value<0 : negative or positive number
- Size[bit] : Size in bit
- Base10 : Decimal(Base10) number.

**See Also**

[print.binary](#)



---

switchEndianess	<i>Switch Endianess.</i>
-----------------	--------------------------

---

**Description**

Switch little-endian to big-endian and vice versa.

**Usage**

```
switchEndianess(x, stickyBits=FALSE)
```

**Arguments**

x	binary number. Any binary number.
stickyBits	Bits wont change if set TRUE. Only the attribute will be switched.

**Value**

switch little-endian to big-endian and vice versa.

**See Also**

[negate](#) or [fillUpToByte](#).

**Examples**

```
x <- as.binary(c(1,1,0,0), logic=TRUE); print(x); summary(x);  
y <- switchEndianess(x); print(y); summary(y);  
y <- switchEndianess(x, stickyBits=TRUE); print(y); summary(y);
```

# Index

as.binary, [2](#), [5](#), [11](#)

bin2gray, [3](#), [11](#)  
binAdd, [4](#)  
binary, [3](#), [4](#), [6](#), [8](#), [11](#)  
binaryLogic, [6](#)  
binaryLogic-package (binaryLogic), [6](#)  
binaryPrefix, [6](#), [7](#), [8](#), [9](#)  
binSeq, [7](#)  
byte, [6](#), [7](#), [8](#), [9](#)  
bytesNeeded, [6-8](#), [9](#), [10](#)

fillUpToBit, [6](#), [9](#), [10](#)  
fillUpToByte, [6-10](#), [10](#), [12](#), [17](#)

gray2bin, [3](#), [11](#)

is.binary, [3](#), [5](#), [11](#)

loadAttributes, [12](#)

negate, [6](#), [10](#), [12](#), [17](#)

Ops.binary, [13](#)

print.binary, [13](#), [16](#)

rotate, [6](#), [14](#), [15](#), [16](#)

saveAttributes, [14](#)  
shiftLeft, [6](#), [14](#), [15](#), [16](#)  
shiftRight, [6](#), [14](#), [15](#), [15](#)  
summary.binary, [13](#), [16](#)  
switchEndianess, [6](#), [10](#), [12](#), [17](#)