Package: bigBits (via r-universe)

September 27, 2024

Type Package

Title Perform Boolean Operations on Large Numbers

Version 1.3

Date 2024-06-27

Description A set of Boolean operators which accept integers of any size, in any base from 2 to 36, including 2's complement format, and perform actions like ``AND,'' ``OR'', ``NOT'', ``SHIFTR/L'' etc. The output can be in any base specified. A direct base to base converter is included.

License LGPL-3

Imports Rmpfr, gmp, methods

NeedsCompilation no

Author Carl Witthoft [aut, cre]

Maintainer Carl Witthoft <cellocgw@gmail.com>

Repository CRAN

Date/Publication 2024-06-28 13:20:03 UTC

Contents

`bigBits-package` *Perform Boolean Operations on Large Numbers*

Description

A set of Boolean operators which accept integers of any size, in any base from 2 to 36, including 2's complement format, and perform actions like "AND," "OR", "NOT", "SHIFTR/L" etc. The output can be in any base specified. A direct base to base converter is included.

Details

The DESCRIPTION file:

Package:	bigBits
Type:	Package
Title:	Perform Boolean Operations on Large Numbers
Version:	1.3
Date:	2024-06-27
Authors@R:	c(person(given = "Carl", family = "Witthoft", email = "cellocgw@gmail.com", role = c("aut", "cre")))
Description:	A set of Boolean operators which accept integers of any size, in any base from 2 to 36, including 2's compleme
License:	LGPL-3
Imports:	Rmpfr, gmp, methods
Author:	Carl Witthoft [aut, cre]
Maintainer:	Carl Witthoft <cellocgw@gmail.com>

Author(s)

Carl Witthoft [aut, cre]

Maintainer: Carl Witthoft <cellocgw@gmail.com>

References

https://en.wikipedia.org/wiki/Two's_complement

See Also

`bitwAnd` and other "bitw*" functions

base2base *Function which converts arbitrary-size integers from any base to any base.*

Description

This function accepts inputs in any base from 2 through 36 and produces the same value in any selected base from 2 through 36. This includes options for signed and 2s complement binary data.

Usage

```
base2base(x,frombase=10, tobase=2, classOut=c('bigz', 'mpfr',
 'numeric','character') , binSize = 0, inTwosComp = FALSE,
   outTwosComp = FALSE )
```

Arguments

x	A value or vector or list of values which are to be converted. The class can generally be numeric, mpfr, bigz, or character strings. Any fractional part is removed, leaving just the integer portion. See Details for more information.
frombase	The base of the input x. If the contents of x are incompatible with the specified base, a warning is issued and that value is skipped (i.e. if x has multiple values, base2base will keep running and process the other values). Default is 10
tobase	The desired base of the output. Default is 2.
classOut	Specify the class of the output. This only has meaning when tobase is 10; all other bases are returned as character strings. Warning: if the input is larger than the max integer size on your system and "numeric" is selected, there will be a roundoff/truncation error.
binSize	Specifies how many digits are to be generated. If this value is less than that necessary to contain the output value, the number of digits will be increased to match. If the output is binary, the final number of digits is expanded to a 4*N value. The default is zero, which allows the function to calculate the minimum bits required. Note: for obvious reasons, this only applies to character-class outputs.
inTwosComp	Only checked if frombase is 2. If FALSE, (the default), the input is positive unless a negative sign is present. If TRUE, the input is handled as a 2s complement binary.
outTwosComp	Only checked if tobase is 2. If FALSE, (the default), the output, if negative, includes a "-" sign. If TRUE, a 2's complement binary.

Details

In general, when submitting an input in other than base 10, it's safest to provide a character string(s). There is some automagical conversion that will take, e.g., a numeric 364 with frombase = 8 and treat as base 8 (thus decimal 244), but this is not guaranteed. Further, keep in mind that numeric

values with more than roughly 16 digits will likely run into floating-point precision errors. For base-10 inputs, use of bigz form is recommended. Inputs in hex format must be character strings. This is because the command parser converts, e.g., 0x3a, to the decimal value 58 prior to passing the value to the function body. Since, as noted above, base2base will attempt to convert a numeric input into the value in the base specified, base2base(0x3a, inbase= 16, ...) will in fact process the input as 58hex, i.e. 88 decimal.

Value

A list containing the converted value(s). Unless tobase is 10, each element is a character string. When tobase is equal to 10, the output class is specified with the argument classOut . Note: if an input or output is incompatible with the specified input or output base, a dummy value "%no" if character, or "NA" if a number-like class, is returned along with a warning message describing the error.

Author(s)

Author and Maintainer:Carl Witthoft <carl@witthoft.com>

See Also

strtoi as.hexmode fracB2B

Examples

```
(base2base(12.4e1,10,16))
(base2base(12.4e-2,10,16))
(base2base(101101,2,10))  # magic. it works!!!
(base2base('1111',2,2,inTwosComp=TRUE, outTwosComp=TRUE))
(base2base('0111',2,2,inTwosComp=TRUE, outTwosComp=TRUE))
(base2base('1111',2,2,inTwosComp=TRUE, outTwosComp=FALSE))
(base2base('0111',2,2,inTwosComp=TRUE, outTwosComp= FALSE))
(base2base(1e55,10,16)) #loses precision before even starting
(base2base('1e55',10,16)) #works
(base2base('1767707668033969' , 10, 36))
```

---

bigAnd *Functions to perform binary operations on integers of arbitrary size, and of arbitrary base (up to 36).*

Description

These functions extend the capabilities of the matching base bitw* functions (which are limited to 32-bit integers). Not only can any integer be processed, at least up to the machine limits as determined with the gmp library, but the inputs and outputs can be in any base. Further, both unsigned (a minus sign indicates negative) and 2s complement base-2 values are allowed.

Usage

```
bigOr(x, y, inBase = 10, outBase = 10, inTwosComp = TRUE)
bigAnd(x, y, inBase = 10, outBase = 10,  inTwosComp = TRUE)
bigXor(x, y, inBase = 10, outBase = 10, inTwosComp = TRUE)
bigNot(x,inBase=10,outBase=10,binSize = 32,inTwosComp = TRUE,outTwosComp = TRUE)
bigShiftL(x, shift = 1,  inBase = 10 , outBase = 10, binSize = 32, inTwosComp = TRUE)
bigShiftR(x, shift = 1,  inBase = 10, outBase = 10, binSize = 32, inTwosComp = TRUE)
bigRotate(x, shift,  inBase = 10,binSize = 32, outBase = 10, inTwosComp = TRUE)
```

Arguments

x, y	The integers to be processed. These can be numeric, integer, mpfr, bigz , or character class. These two items must be of the same class. List variables are acceptable so long as the contents are all of one class. They can be any base from 2 through 36 as specified by inBase. If these are character strings, formats such as "-37e+5" or "0x4e" (for hex data) are accepted. See the Details section for the capabilities and limits on "translation" of inputs. If the lengths of x and y differ, the shorter one will be silently recycled.
inBase	Specify the designated base of the input(s) . Default is 10.
outBase	Specify the designated base of the output(s) . Default is 10.
inTwosComp	When inBase is 2, the input(s) is treated as being in 2s complement format when this is TRUE (the default). Otherwise the input(s) is treated as a positive base-2 value unless a negative sign is present. . If inBase is not equal to 2, this arg is ignored.
binSize	Specify the number of binary bits for the output calculation. If this is set to zero (the default), the minimum number is set to 4*N such that the current value of the input and output is containable. But see the Details section for a discussion of 2s complement behavior.
outTwosComp	Whent outBase is 2, return the 2s complement version of the value(s). Default is TRUE; when False, return a binary value(s) with a negative sign as necessary.
shift	The number of bits to shift the input by. Only positive values are allowed for bigShiftL and bigShiftR, which shift to the "left" and "right" respectively. bigRotate accepts positive or negative values and rotates in the prescribed direction accordingly. See the Details section for comments on 2s complement inputs.

Details

The inputs, when not in base 10, are expected to follow the common encoding where the letters "a" through "z" correspond to the decimal values 10 through 35. Values in bases greater than 10 must be character strings. If the input is base 16 ('hex'), the character string can begin with or without '0x'. Inputs specified as base 2 through 10 can be provided in any of the numeric formats and the functions automagically interpret them correctly. For example, when x is numeric 1101 and inBase is 2 , the functions will interpret the input as 13 if inTwosComp is FALSE and as -3 if TRUE.

Shifting to the right when 2s complement is in use can lead to unexpected results. bigShiftR assumes 32-bit binary 2scomp for compatibility with bitwShiftR. But for an arbitrarily large

binary 2s complement input, the output, for a shift of one, will move -1 (11111...) to 2^(N-1) -1 , where N is the number of bits including the sign bit. `bigShiftR` defaults to max(32, min_needed_for_magnitude_of_x) bits. Similarly, `bigShiftL` by default provides sufficient bits to handle the shifted value. This is unlike `bitwShiftL` which returns the value of the 32 LSBs (in 2s complement form) if the shifted value exceeds 2^31-1 . If `binSize` is not zero (the default), `bigShiftL` will truncate to the specified bit size (or 32, whichever is greater).

`bigRotate` converts input 2s complement binaries to unsigned binaries (with a negative sign when needed). This is because the behavior of different compilers with respect to rotating 2s complement binary data can be different or even unspecified. When the input is negative (in any base), the rotation is applied to the positive unsigned binary equivalent and a negative sign attached to the output. In particular, this means that 2s complement output is disallowed.

Note that, for compability with the base `bitw*` functions, the value is internally extended to (at least) 32 bits prior to bitwise operations. In particular, the value of the NOT function when 2s complement is in use depends on the specified size of the binary data. Remember that there will be precision errors if large numerics are entered, possibly leading to roundoff errors. In general, it is safer to enter values in `bigz` format or as character strings.

Value

A list object with one value per entry, corresponding to the input value(s) of x (or y if y is the longer input). In most cases the entries are character strings. However, if the input and the output are specified as base 10, then the output is converted to the class of the input.

Author(s)

Author and Maintainer:Carl Witthoft `<carl@witthoft.com>`

See Also

`bitwAnd` and other "bitw*" functions

---

buildBinaries *Function to convert values to binary form*

Description

This function is intended primarily for internal use by the `big*` Boolean functions. Its job is to take an input in any base, in almost any class (numeric, character, etc) and generate the binary form of the same value.

Usage

```
buildBinaries(x, y= NULL,inBase, inTwosComp = FALSE, binSize = 32)
```

Arguments

x, y	The values to be converted. Typically these are the x,y values provided to one of the Boolean functions in this package. If only x is input, y defaults to NULL
inBase	The base (2 thru 36) of the input values.
inTwosComp	When the input inBase is 2, this specifies whether the input is positive unless a negative sign is present, or 2's complement format .
binSize	The minimum number of bits to use for the output binary data. If insufficient for the size of the input(s), this will be increased to the next 4*N size. When there are two inputs, both outputs are set to the same number of bits. ~

Value

xbin	A vector of numeric ones and zeroes representing the binary form of x
ybin	A vector of numeric ones and zeroes representing the binary form of y. If the input is NULL, a single value of 0 is returned.

Author(s)

Author and Maintainer:Carl Witthoft <carl@witthoft.com>

Examples

```
buildBinaries(73,-73,inBase=10)
```

---

fracB2B — *Function which converts fractions (between 1 and 0) from any base to any other base.*

Description

This function accepts inputs in any base from 2 through 36 and generates the fractional portion of the input values in any selected base from 2 through 36.

Usage

```
fracB2B( x, inBase = 10, outBase = 16, maxdig = 0)
```

Arguments

x	A value or vector or list of values which are to be converted. The class can generally be numeric, mpfr,bigz,bigq, or character strings. Any integer part is removed, as only the fractional part is converted with this function. See Details for more information.

inBase	The base of the input x, in the range 2 through 36. If the contents of x are incompatible with the specified base, a warning is issued and that value is skipped (i.e. if x has multiple values, base2base will keep running and process the other values). Default is 10.
outBase	The desired base of the output, in the range 2 through 36. Default is 16.
maxdig	The maximum number of digits to return in each result. This avoids an infinite loop when a given decimal does not terminate in the output base. The default value of 0 causes the function to generate a "reasonable" estimate for the number of places needed to maintain precision.

Details

In general, it's safest to provide a character string(s). There is some automagical conversion that will take, e.g., a numeric 364 with inBase = 8 and treat as base 8 (thus decimal 244), but this is not guaranteed. Further, keep in mind that numeric values with more than roughly 16 digits will likely run into floating-point precision errors. Inputs in any base greater than 10 must be character strings. This is because the command parser converts, e.g., 0x3a, to the decimal value 58 prior to passing the value to the function body.

Value

A vector containing the converted value(s) as strings. A negative sign is included for negative inputs.

Author(s)

Author and Maintainer:Carl Witthoft <carl@witthoft.com>

See Also

base2base

Examples

```
(fracB2B(12.43e2,10,16)) # no decimal part
(fracB2B(12.43e-2,10,16))
(fracB2B(101.101,2,10)) # magic. it works!!!
fracB2B('.357') # "0.5b64"
fracB2B('.357',maxdig = 10) #"0.5b645a1cac"
fracB2B(".5b64",16,10)
fracB2B(".5b645a1cac",16,10)
```

noExp *Function to convert character-string numbers in exponential notation to "pure" integers.*

Description

This is a helper function for base2base. When an input value is a character string with exponential notation, e.g., "2.65e4" , this function rebuilds the character string as a pure integer, e.g., "26500" . Decimal portions are removed.

Usage

```
noExp(x)
```

Arguments

x	An input character string, assumed only to contain numerals 0-9, "+,-,e,E" and the decimal separator character defined in the current locale.

Details

Both input and output must be base 10, as exponentiation in other bases is outside the current scope of this package.

Value

A character string representing the input as an integer written "longhand," i.e. no exponent. This string will contain only numerals and possibly a lead minus sign in the case of negative inputs.

Author(s)

Author and Maintainer:Carl Witthoft <carl@witthoft.com>

Examples

```
noExp('37e3')
noExp('-2.345e4')
# this returns zero
noExp('234e-5')
```

Index