rgates vinette

Muhammad Ezzat

3/12/2020

Introduction

Digital Logic designers build complex electronic components that use both electrical and computational characteristics. These characteristics may involve power, current, logical function, protocol and user input. Digital Logic Design is used to develop hardware, such as circuit boards and microchip processors.

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a certain logic. Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

The rgates package

The rgates package aim to simulate logic gates for 2 variables. In fact the number of possible boolean fucntions that can be implemented on logic gates is 2 raised to the power of double the number of values. In our case its 2^(2*2) which evaluates to 16. On first thought this needs 16 fucntions to represent, but actually somw fucntions are input order sensitive so a change in input order is considered a new fucntion. Thus the number of functions is reduced to 12 instead of 16. However 2 helper functions were created in addition to 1 complementary function so we have 15 functions in total.

Tutorial

Let's walk through the package and implement the rgates package fucntions. First we will create two logical vectors that form all of the possible logical combinations through the truth_table() function.

```
df <- truth_table()
print(df)
```

```
##       x     y
## 1 FALSE FALSE
## 2 FALSE  TRUE
## 3  TRUE FALSE
## 4  TRUE  TRUE
```

As you saw it creates a data frame of two vectors representing the two variables that include all the possible logical combinations to get. Now let's explore our gates.

Not gate

Also known as the inverter it returns false from a true input & vise versa.

```
print(df$x)

## [1] FALSE FALSE  TRUE  TRUE

print(not(df$x))

## [1]  TRUE  TRUE FALSE FALSE
```

And gate

The equivalence of the '&' sign in R & most of the programming languages, returns true only if both inputs are so.

```
print(df$x)

## [1] FALSE FALSE  TRUE  TRUE

print(df$y)

## [1] FALSE  TRUE FALSE  TRUE

print(and(df$x,df$y))

## [1] FALSE FALSE FALSE  TRUE
```

Or gate

The equivalence of the '|' sign in R & most of the programming languages, returns false only if both inputs are so.

```
print(df$x)

## [1] FALSE FALSE  TRUE  TRUE

print(df$y)

## [1] FALSE  TRUE FALSE  TRUE

print(or(df$x,df$y))

## [1] FALSE  TRUE  TRUE  TRUE
```

Nand gate

Simply the inverse of the and gate, you can write x nand y as !(x&y). We will confirm this right at the moment using rgates package.

```
print(df$x)

## [1] FALSE FALSE  TRUE  TRUE

print(df$y)

## [1] FALSE  TRUE FALSE  TRUE

print(nand(df$x,df$y))

## [1]  TRUE  TRUE  TRUE FALSE

print(not(and(df$x,df$y)))

## [1]  TRUE  TRUE  TRUE FALSE
```

Nor gate

Similarly, the nor gate is the inverse of the or gate.

```
print(df$x)

## [1] FALSE FALSE  TRUE  TRUE

print(df$y)

## [1] FALSE  TRUE FALSE  TRUE

print(nor(df$x,df$y))

## [1]  TRUE FALSE FALSE FALSE

print(not(or(df$x,df$y)))

## [1]  TRUE FALSE FALSE FALSE
```

Xor gate

XOR gate (sometimes EOR, or EXOR and pronounced as Exclusive OR) is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is odd. An XOR gate implements an exclusive or; that is, a true output results if one, and only one, of the inputs to the gate is true.

```
print(df$x)

## [1] FALSE FALSE  TRUE  TRUE
```

```
print(df$y)
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```
print(xor(df$x,df$y))
```

```
## [1] FALSE  TRUE  TRUE FALSE
```

Xnor gate

Mathematically it represents the equivellance state, returns true if both inputs are of the same value.

```
print(df$x)
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
print(df$y)
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```
print(xnor(df$x,df$y))
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

Implification gate

An implication is something that is suggested, or happens, indirectly. When you left the gate open and the dog escaped, you were guilty by implication. Usually, when used in the plural, implications are effects or consequences that may happen in the future. However in logic An implication is the compound statement of the form "if p, then q." It is denoted p⇒q, which is read as "p implies q." It is false only when p is true and q is false, and is true in all other situations.

```
print(df$x)
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
print(df$y)
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```
print(implies(df$x,df$y))
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

Inhibition gate

Instead of a long explaination we can just coin the term "X but not Y". So it only returns true if x is but y isn't.

```
print(df$x)
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
print(df$y)
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```
print(inhibit(df$x,df$y))
```

```
## [1] FALSE FALSE  TRUE FALSE
```

BinaryConstant gate

They produce either true or false for whatever the length you need

```
print(on(3))
```

```
## [1] TRUE TRUE TRUE
```

```
print(null(3))
```

```
## [1] FALSE FALSE FALSE
```

Transefer gate

Another very simple gate, baiscally it returns it's input, it's usage in real life is a sort of amplifier when the curcuit power loss effect appears on your signals. It was provided by the rgates package for convenience.

```
print(df$x)
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
print(transefer(df$x))
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

Chaining

You probably noticed that you can chain many gates together, here is an example

```
print(implies(and(df$y,xor(df$x,on(4))),null(4)))
```

```
## [1]  TRUE FALSE  TRUE  TRUE
```

Warnings

All of these functions throw errors for non logical type input and for inputs not equal in length. This package is compeletly safe to use as it passed the TRAVIS CL test & RStudio test. The package is under the CC0 license which means anybody can use it for free. If you

have any issue contact the maintainer : muhammadezzat316@gmail.com. Github repo : https://github.com/MuhammadEzzatHBK/rgates2 .