Let us start the year with something for your inner maths nerd 

Source: Wikimedia

For those of you who don’t yet know Rosetta Code: it is a real cool site where you can find lots of interesting code examples in all kinds of different languages for many different tasks. Of course R is also present big time (at the time of writing 426 code examples!): [Rosetta Code for R](https://rosettacode.org/wiki/Category:R).

The name of the site is inspired by the famous [Rosetta Stone](https://en.wikipedia.org/wiki/Rosetta_Stone) of Ancient Egypt which is inscribed with three different versions of the same text: in Ancient Egyptian hieroglyphs, Demotic script, and Ancient Greek script which proved invaluable in deciphering Egyptian hieroglyphs and thereby opening the window into ancient Egyptian history.

Now, a few days a ago I again added an example (for the other tasks I solved I will write more posts in the future, so stay tuned!). The task is to verify the correctness of Machin-like formulae using exact arithmetic.

A little bit of mathematical background is in order, so [Wikipedia](https://en.wikipedia.org/wiki/Machin-like_formula) to the rescue:

Machin-like formulae are a popular technique for computing \pito a large number of digits. They are generalizations of John Machin]s formula from 1706:

\[\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}\]

which he used to compute \pito 100 decimal places.

Machin-like formulae have the form

\[c_0 \frac{\pi}{4} = \sum_{n=1}^N c_n \arctan \frac{a_n}{b_n}\]

where a_nand b_nare positive integers such that a_n < b_n, c_nis a signed non-zero integer, and c_0is a positive integer.

The exact task is to verify that the following Machin-like formulae are correct by calculating the value of tan (right hand side) for each equation using exact arithmetic and showing they equal one:

{\pi\over4} = \arctan{1\over2} + \arctan{1\over3}  
{\pi\over4} = 2 \arctan{1\over3} + \arctan{1\over7}  
{\pi\over4} = 4 \arctan{1\over5} - \arctan{1\over239}  
{\pi\over4} = 5 \arctan{1\over7} + 2 \arctan{3\over79}  
{\pi\over4} = 5 \arctan{29\over278} + 7 \arctan{3\over79}  
{\pi\over4} = \arctan{1\over2} + \arctan{1\over5} + \arctan{1\over8}  
{\pi\over4} = 4 \arctan{1\over5} - \arctan{1\over70} + \arctan{1\over99}  
{\pi\over4} = 5 \arctan{1\over7} + 4 \arctan{1\over53} + 2 \arctan{1\over4443}  
{\pi\over4} = 6 \arctan{1\over8} + 2 \arctan{1\over57} + \arctan{1\over239}  
{\pi\over4} = 8 \arctan{1\over10} - \arctan{1\over239} - 4 \arctan{1\over515}  
{\pi\over4} = 12 \arctan{1\over18} + 8 \arctan{1\over57} - 5 \arctan{1\over239}  
{\pi\over4} = 16 \arctan{1\over21} + 3 \arctan{1\over239} + 4 \arctan{3\over1042}  
{\pi\over4} = 22 \arctan{1\over28} + 2 \arctan{1\over443} - 5 \arctan{1\over1393} - 10 \arctan{1\over11018}  
{\pi\over4} = 22 \arctan{1\over38} + 17 \arctan{7\over601} + 10 \arctan{7\over8149}  
{\pi\over4} = 44 \arctan{1\over57} + 7 \arctan{1\over239} - 12 \arctan{1\over682} + 24 \arctan{1\over12943}

The same should be done for the last and most complicated case…

{\pi\over4} = 88 \arctan{1\over172} + 51 \arctan{1\over239} + 32 \arctan{1\over682} + 44 \arctan{1\over5357} + 68 \arctan{1\over12943}

… but it should be confirmed that the following, slightly changed, formula is incorrect by showing tan (right hand side) is **not** one:

{\pi\over4} = 88 \arctan{1\over172} + 51 \arctan{1\over239} + 32 \arctan{1\over682} + 44 \arctan{1\over5357} + 68 \arctan{1\over12944}

This is what I contributed to [Rosetta Code](https://rosettacode.org/wiki/Check_Machin-like_formulas#R):

library(Rmpfr)

prec <- 1000 # precision in bits

`%:%` <- function(e1, e2) '/'(mpfr(e1, prec), mpfr(e2, prec)) # operator %:% for high precision division

# function for checking identity of tan of expression and 1, making use of high precision division operator %:%

tanident\_1 <- function(x) identical(round(tan(eval(parse(text = gsub("/", "%:%", deparse(substitute(x)))))), (prec/10)), mpfr(1, prec))

tanident\_1( 1\*atan(1/2) + 1\*atan(1/3) )

## [1] TRUE

tanident\_1( 2\*atan(1/3) + 1\*atan(1/7))

## [1] TRUE

tanident\_1( 4\*atan(1/5) + -1\*atan(1/239))

## [1] TRUE

tanident\_1( 5\*atan(1/7) + 2\*atan(3/79))

## [1] TRUE

tanident\_1( 5\*atan(29/278) + 7\*atan(3/79))

## [1] TRUE

tanident\_1( 1\*atan(1/2) + 1\*atan(1/5) + 1\*atan(1/8) )

## [1] TRUE

tanident\_1( 4\*atan(1/5) + -1\*atan(1/70) + 1\*atan(1/99) )

## [1] TRUE

tanident\_1( 5\*atan(1/7) + 4\*atan(1/53) + 2\*atan(1/4443))

## [1] TRUE

tanident\_1( 6\*atan(1/8) + 2\*atan(1/57) + 1\*atan(1/239))

## [1] TRUE

tanident\_1( 8\*atan(1/10) + -1\*atan(1/239) + -4\*atan(1/515))

## [1] TRUE

tanident\_1(12\*atan(1/18) + 8\*atan(1/57) + -5\*atan(1/239))

## [1] TRUE

tanident\_1(16\*atan(1/21) + 3\*atan(1/239) + 4\*atan(3/1042))

## [1] TRUE

tanident\_1(22\*atan(1/28) + 2\*atan(1/443) + -5\*atan(1/1393) + -10\*atan(1/11018))

## [1] TRUE

tanident\_1(22\*atan(1/38) + 17\*atan(7/601) + 10\*atan(7/8149))

## [1] TRUE

tanident\_1(44\*atan(1/57) + 7\*atan(1/239) + -12\*atan(1/682) + 24\*atan(1/12943))

## [1] TRUE

tanident\_1(88\*atan(1/172) + 51\*atan(1/239) + 32\*atan(1/682) + 44\*atan(1/5357) + 68\*atan(1/12943))

## [1] TRUE

tanident\_1(88\*atan(1/172) + 51\*atan(1/239) + 32\*atan(1/682) + 44\*atan(1/5357) + 68\*atan(1/12944))

## [1] FALSE

As you can see all statements are TRUE except for the last one!

In the code I make use of the Rmpfr package (from Martin Maechler of ETH Zürich, Switzerland) which is based on the excellent GMP (GNU Multiple Precision) library. I define a new infix operator %:% for high-precision division and after that convert all standard divisions in the formulae to high-precision divisions and calculate the tan. Before I check if the result is identical to one I round it to 100 decimal places which is more than enough given the precision of log_{10}(2^{1000})=301.03, so about 300 decimal places, in the example.

Please let me know in the comments what you think of this approach and whether you see room for improvement for the code – Thank you!