Model output test

Introduction

Currently, we're using a temporary method in order to store the model output before processing them. This method is a list which has two levels. The first level is the DoE level and the second is the Usm which contains a tibble with the variables and their values.

We wanted to know if a different storage method was more adapted to our problem. So, we have implemented a second storage method which consists in a single tibble which contains all the information for each DoE and each Usm.

There are two parts in this document. The first is a description part where we'll print the storage methods, in order to make you understand how it is store inside the differents methods. The second part is the tests part where we first describe the functions used for extraction and their use cases. And then, the results of the extraction tests for each use case.

Methods

Let's print the storage methods. For this example, the methods contain 2 DoE's level and 6 Usms.

First method : List

```
# DoE, usm_number
li <- create_list(USM_list_1996,2,6,sim_data_5)
li
```

```
## $`1`
## $`1`$`bo96iN+_1`
## # A tibble: 5 x 11
##   Date                  jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>              <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  31.6  28.6  23.0     0     0
## 2 1996-01-02 00:00:00     2     0       0       0  31.6  28.6  23.0     0     0
## 3 1996-01-03 00:00:00     3     0       0       0  31.6  28.6  23.1     0     0
## 4 1996-01-04 00:00:00     4     0       0       0  31.6  28.6  23.7     0     0
## 5 1996-01-05 00:00:00     5     0       0       0  31.6  28.6  26.0     0     0
## # ... with 1 more variable: resmes <dbl>
##
## $`1`$`bo96iN+_2`
## # A tibble: 5 x 11
##   Date                  jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>              <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  31.6  28.6  23.0     0     0
```

```
## 2 1996-01-02 00:00:00     2     0       0       0  31.6  28.6  23.0     0     0
## 3 1996-01-03 00:00:00     3     0       0       0  31.6  28.6  23.1     0     0
## 4 1996-01-04 00:00:00     4     0       0       0  31.6  28.6  23.7     0     0
## 5 1996-01-05 00:00:00     5     0       0       0  31.6  28.6  26.0     0     0
## # ... with 1 more variable: resmes <dbl>
##
## $`1`$`lu96iN+_1`
## # A tibble: 5 x 11
##   Date                  jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>              <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
## $`1`$`lu96iN+_2`
## # A tibble: 5 x 11
##   Date                  jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>              <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
## $`1`$lu96iN6_1
## # A tibble: 5 x 11
##   Date                  jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>              <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
## $`1`$lu96iN6_2
## # A tibble: 5 x 11
##   Date                  jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>              <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
##
## $`2`
## $`2`$`bo96iN+_1`
## # A tibble: 5 x 11
```

```
##   Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  31.6  28.6  23.0     0     0
## 2 1996-01-02 00:00:00     2     0       0       0  31.6  28.6  23.0     0     0
## 3 1996-01-03 00:00:00     3     0       0       0  31.6  28.6  23.1     0     0
## 4 1996-01-04 00:00:00     4     0       0       0  31.6  28.6  23.7     0     0
## 5 1996-01-05 00:00:00     5     0       0       0  31.6  28.6  26.0     0     0
## # ... with 1 more variable: resmes <dbl>
##
## $`2`$`bo96iN+_2`
## # A tibble: 5 x 11
##   Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  31.6  28.6  23.0     0     0
## 2 1996-01-02 00:00:00     2     0       0       0  31.6  28.6  23.0     0     0
## 3 1996-01-03 00:00:00     3     0       0       0  31.6  28.6  23.1     0     0
## 4 1996-01-04 00:00:00     4     0       0       0  31.6  28.6  23.7     0     0
## 5 1996-01-05 00:00:00     5     0       0       0  31.6  28.6  26.0     0     0
## # ... with 1 more variable: resmes <dbl>
##
## $`2`$`lu96iN+_1`
## # A tibble: 5 x 11
##   Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
## $`2`$`lu96iN+_2`
## # A tibble: 5 x 11
##   Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
## $`2`$lu96iN6_1
## # A tibble: 5 x 11
##   Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
##
## $`2`$lu96iN6_2
```

```
## # A tibble: 5 x 11
##   Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4  HR_5
##   <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1996-01-01 00:00:00     1     0       0       0  22.5  25.4  23.7  26.8     0
## 2 1996-01-02 00:00:00     2     0       0       0  22.5  25.4  23.7  26.8     0
## 3 1996-01-03 00:00:00     3     0       0       0  22.5  25.4  23.8  26.8     0
## 4 1996-01-04 00:00:00     4     0       0       0  22.5  25.4  24.4  26.8     0
## 5 1996-01-05 00:00:00     5     0       0       0  22.5  25.4  25.8  28.4     0
## # ... with 1 more variable: resmes <dbl>
```

Second method : Tibble

```
# DoE, usm_number
tb <- create_tibble(USM_list_1996,2,6,sim_data_5)
tb
```

```
## # A tibble: 60 x 13
##    Name  Date                 jul lai_n masec_n mafruit  HR_1  HR_2  HR_3  HR_4
##    <chr> <dttm>             <int> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 bo96~ 1996-01-01 00:00:00     1     0       0       0  31.6  28.6  23.0     0
##  2 bo96~ 1996-01-02 00:00:00     2     0       0       0  31.6  28.6  23.0     0
##  3 bo96~ 1996-01-03 00:00:00     3     0       0       0  31.6  28.6  23.1     0
##  4 bo96~ 1996-01-04 00:00:00     4     0       0       0  31.6  28.6  23.7     0
##  5 bo96~ 1996-01-05 00:00:00     5     0       0       0  31.6  28.6  26.0     0
##  6 bo96~ 1996-01-01 00:00:00     1     0       0       0  31.6  28.6  23.0     0
##  7 bo96~ 1996-01-02 00:00:00     2     0       0       0  31.6  28.6  23.0     0
##  8 bo96~ 1996-01-03 00:00:00     3     0       0       0  31.6  28.6  23.1     0
##  9 bo96~ 1996-01-04 00:00:00     4     0       0       0  31.6  28.6  23.7     0
## 10 bo96~ 1996-01-05 00:00:00     5     0       0       0  31.6  28.6  26.0     0
## # ... with 50 more rows, and 3 more variables: HR_5 <dbl>, resmes <dbl>,
## #   DoE <int>
```

Tests

Use cases

First, let us describe the use cases. There are 3 use cases : Optimization, Multi-Simulation and Analysis. Each use case has one function matching with the storage type, so each use case has two functions, one per storage method.

Now, let's see what returns each function and how they are called.

Optimization's use case

We're beginning with the optimization's use case. This function returns all the values for a variable with the dates. It takes as parameters a DoE level, a Usm name and a variable name.

```
res <- tibble_get_dates_and_var_values(tb,2,"bo96iN+_2","HR_1")
res
```

```
## # A tibble: 5 x 2
##   Date                 HR_1
##   <dttm>              <dbl>
## 1 1996-01-01 00:00:00  31.6
## 2 1996-01-02 00:00:00  31.6
## 3 1996-01-03 00:00:00  31.6
## 4 1996-01-04 00:00:00  31.6
## 5 1996-01-05 00:00:00  31.6
```

```
li2 <- list_get_dates_and_var_values(li,2,"bo96iN+_2","HR_1")
li2
```

```
## # A tibble: 5 x 2
##   Date                 HR_1
##   <dttm>              <dbl>
## 1 1996-01-01 00:00:00  31.6
## 2 1996-01-02 00:00:00  31.6
## 3 1996-01-03 00:00:00  31.6
## 4 1996-01-04 00:00:00  31.6
## 5 1996-01-05 00:00:00  31.6
```

Multi-simulations' use case

This function returns the Usms names with the variable's values. It takes as parameters a DoE level, a variable name and a date.

```
res <- tibble_get_usm_names_and_var_values(tb,2,"HR_4","1996-01-05")
res
```

```
## # A tibble: 6 x 2
##   Name       HR_4
##   <chr>     <dbl>
## 1 bo96iN+_1   0
## 2 bo96iN+_2   0
## 3 lu96iN+_1  28.4
## 4 lu96iN+_2  28.4
## 5 lu96iN6_1  28.4
## 6 lu96iN6_2  28.4
```

```
li2 <- list_get_usm_names_and_var_values(li,2,"HR_4","1996/01/05")
li2
```

```
##      names       values
## [1,] "bo96iN+_1" 0
## [2,] "bo96iN+_2" 0
## [3,] "lu96iN+_1" 28.35693
## [4,] "lu96iN+_2" 28.35693
## [5,] "lu96iN6_1" 28.35693
## [6,] "lu96iN6_2" 28.35693
```

Analysis' use case

This function returns the DoE levels and the variable's values. It takes as parameters an Usm name, a variable name and a date.

```
res <- tibble_get_DOE_and_var_values(tb,"bo96iN+_2","HR_3","1996-01-05")
res
```

```
## # A tibble: 2 x 2
##     DoE  HR_3
##   <int> <dbl>
## 1     1  26.0
## 2     2  26.0
```

```
li2 <- list_get_DOE_and_var_values(li,"bo96iN+_2","HR_3","1996/01/05")
li2
```

```
##      DOE values
## [1,] 1   26.00001
## [2,] 2   26.00001
```

Tests' environment

Work station

The following tests have been realised on the Windows OS 64 bit version. The R memory allocated initially was 8143 bytes. For some tests, the memory was raised to 8To via the memory.limit(size=) function. The test on the Windows OS were done using an Intel Core i5-8400 CPU 2.80GHz processor and a 8Go Ram. The test on Linux OS were done using a 16 core processor, 62.7Go Ram, 134Go swap, Ubuntu 18.04.4 LTS 64bit.

Protocole

All the tests have been realised on a R script. The R memory was cleared between tests by deleting unused variables and calling the garbage collector. Also, the only active program on the computer was RStudio, in order to prevent disturbances during the tests.

Maximum data generated

List All the structures generated below didn't need any memory boost.

Tibble First, for the optimization's use case with 20 dates, the maximum DoE level we can generate with 100 Usms is 130K but we can't process it. Instead, the maximum DoE level we can process is 120K, with a memory boost. Without, we can generate a 40K DoE level. In the case with 289 dates and 10 Usms, the maximum DoE level generated is 50K with boost and 20K without.

Then, for the multi-simulation's use case with 5 dates, the maximum Usms we can generate with a single DoE level is 20M with memory boost and 7M500K Usms without. In the case we have 289 dates, the maximum generated is 600K Usms but we can only process 500K Usms with memory boost and without we can generate 140K Usms.

Finally, for the analysis' use case with 5 dates and 10 Usms, we can generate a 4M DoE level with boost and a 1M DoE level without. In the case we have 289 dates, the maximum level DoE generated is 50K with boost and without it is 20K.

Functions

Creation We have tested 3 creation functions for the tibble and 2 for the list.

For the tibble, the first creation function use a pre allocated tibble. The second function and the third are without pre allocation. The last one, is not build the same way as the others, the tibble is organized by usm then by DoE which is the contrary of the previous one. The third function is designed to gain memory weight and allows us to generate tibble with higher dimensions than the previous functions. The last creation function is also designed for an extraction test to be faster due to the way it is build.

As results, the third function doesn't meet our expectations, it doesn't allow us to create tibble with higher dimensions and it is not faster on the extraction tests.

For the list, the first creation function is a pre allocated function and the second is a non pre allocated function.

Optimization's use case For this extraction test, we have 2 functions for the tibble. The first uses the functions of the dplyr package, while the second uses the subset function. The second function is faster, because it have less operation to do.

Multi-simulation's use case For this extraction test, we have 2 functions for the list. The first function returns a pre allocated list while the second returns a non pre allocated tibble. The second function is faster because the pre allocation takes too much time.

Moreover, the second function had 2 versions, one with a binding on the name at the end and one with the binding on the name at the beginning. The results are clear, the function with binding on the name at the beginning is faster than the other.

Analysis' use case This extraction test is the same as the previous. It has 2 functions for the list. The two functions have the same specificities as the multi-simulation's use case. The results are exactly the same as the multi-simulation's use case.

Creation's tests

Now it is time for the instanciations' tests of each storage method. However, as the current method is only temporary and we are not sure if the second will be the one that suits us the most, the instanciations' tests are not very important at the moment. If you want some details on it, you can go on the Rmd file and delete the echo=FALSE on each chunk and generate again the document.

Extraction's tests

We arrive now at the most important part, the extraction's tests. In this part, we will compare the results of each storage method on each use case.

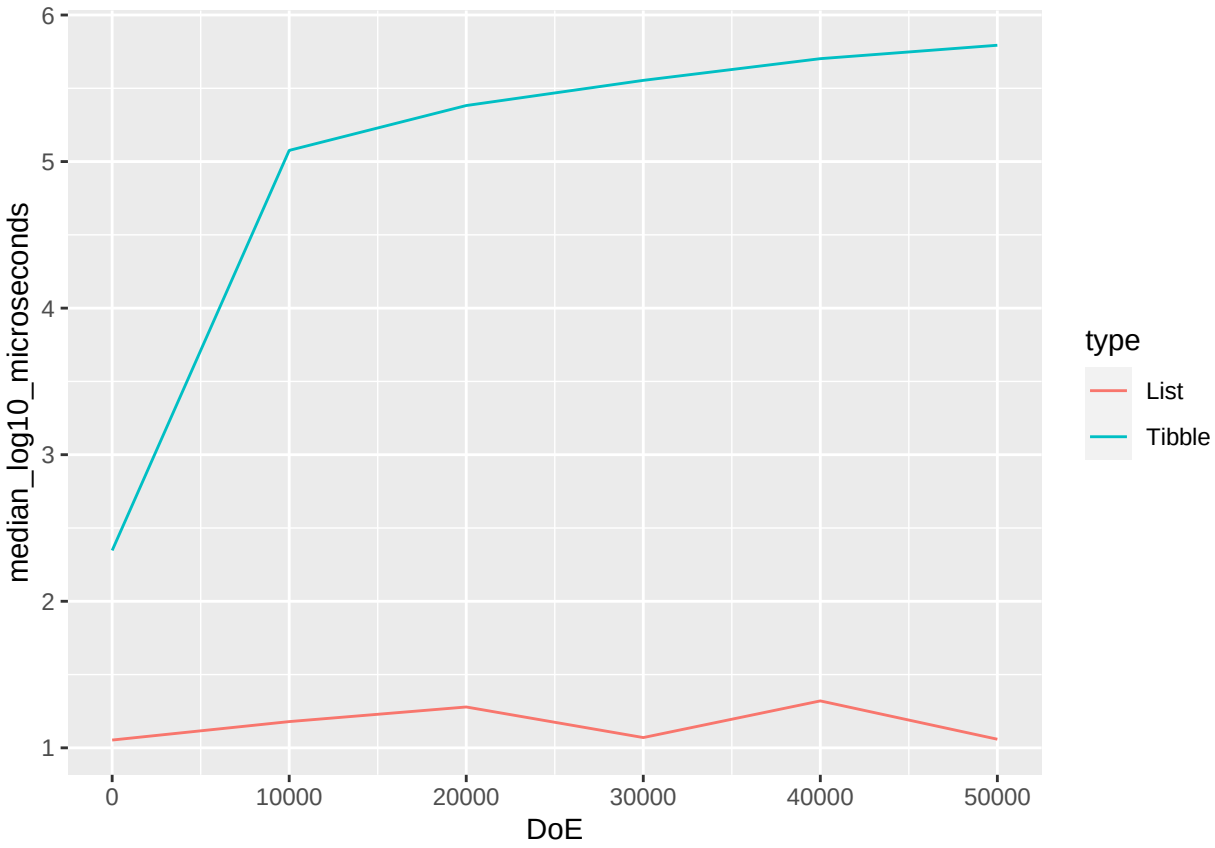
For each use case's tests, optimization, multisimulation and analysis, we have done two differents types of tests. The first one, which is called "first setup" is based on the maximum of DoE and Usms we can generate and process through the functions, on the windows OS. The second type of test, called "second setup" consists on the tests that are usually made using the functions.

Each type of test are made with some complementary tests in order to make graphics on the functions' execution time.

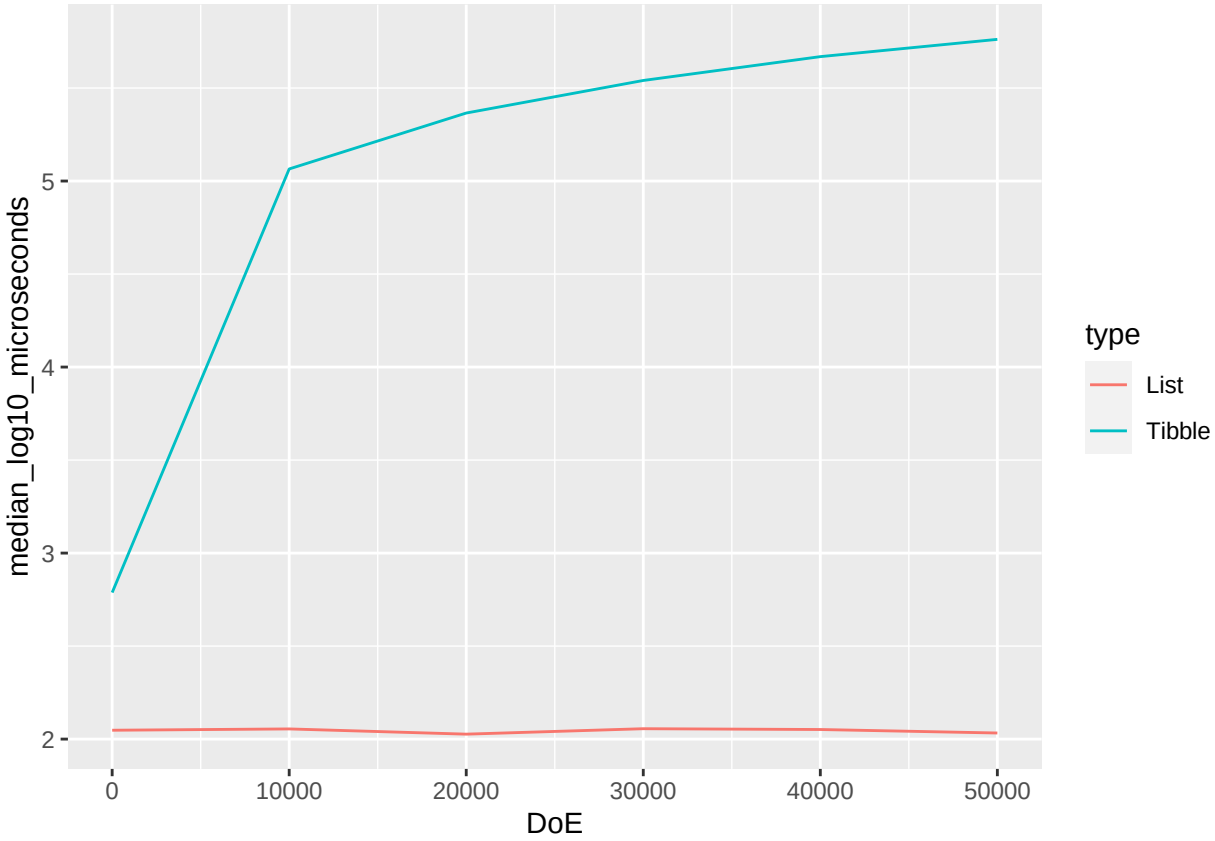
Optimization's extraction test

FOr this test, we get the dates along with variable's values that we want for a specific DoE ans USm.

First setup : DoE from 1 to 50K, 10 Usms and 289 Dates Here are the results for the tests on Windows OS :

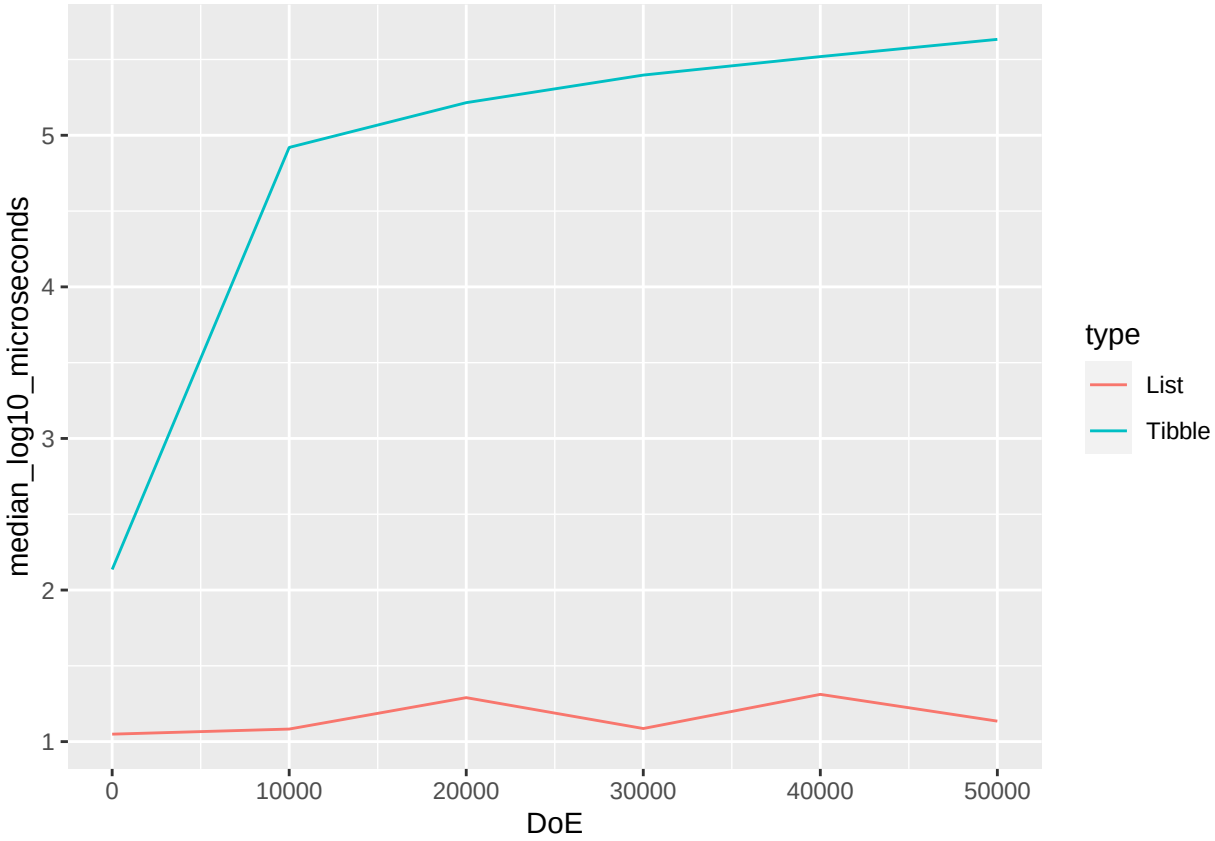


Here are the results for the tests on Linux OS :

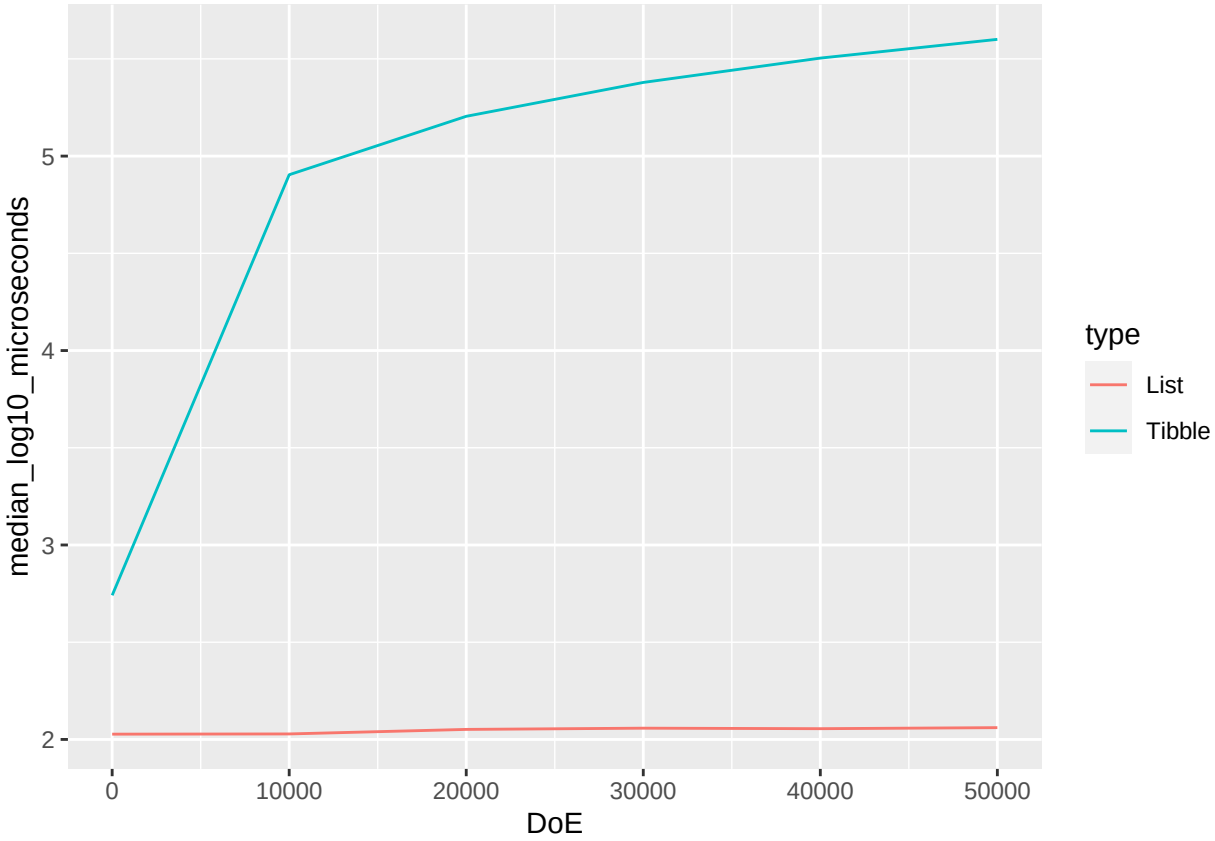


As we can see, the List type clearly outperforms the Tibble type.

Second setup : DoE from 1 to 50K, 100 Usms and 20 Dates Windows OS results :



Linux OS results :



Once again, the List type is clearly better than the Tibble Type.

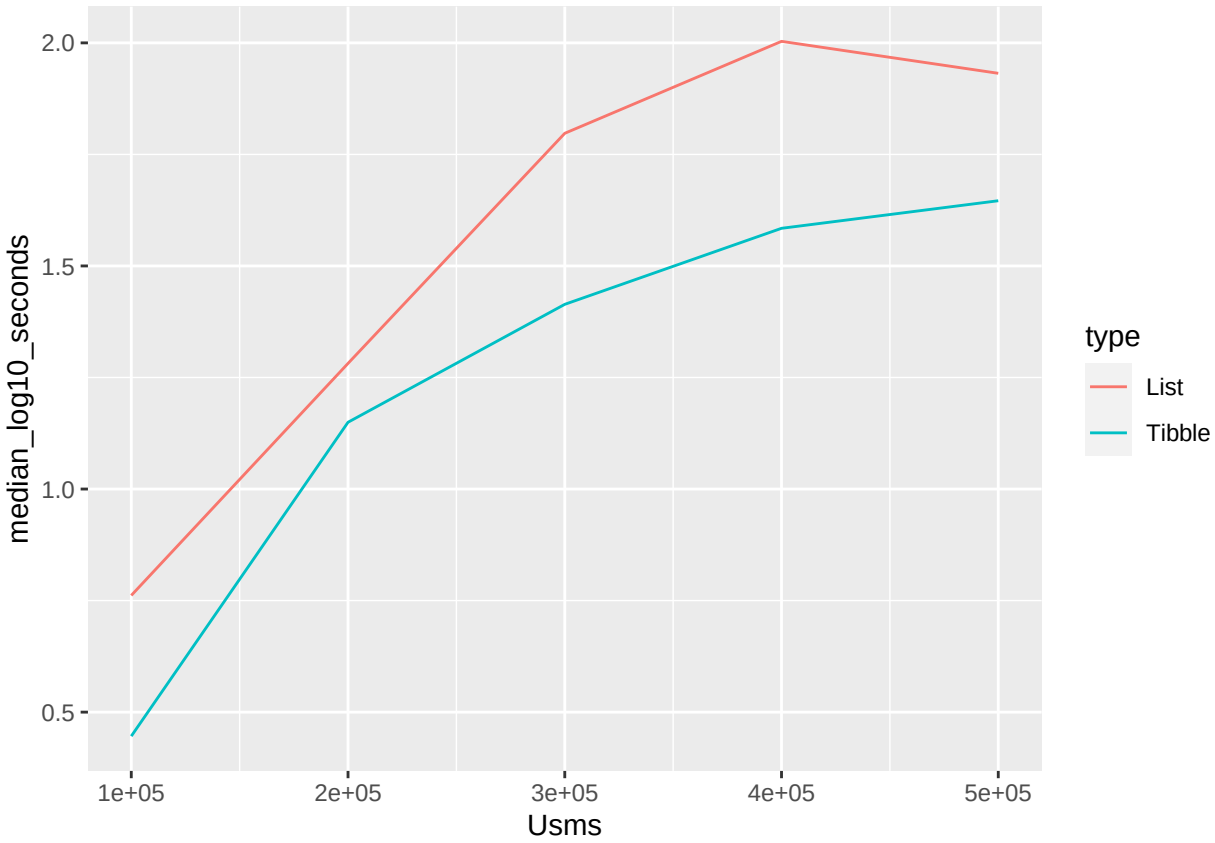
We can conlude by saying this, for the opmization's use case, the List type is clearly the most suited of the two. No matter is we use the optimization's use case's functions on the maximum we could process on windows OS way or on the usual use.

This can be explained by the fact that is the tibble function, there are more operation done than in the List function.

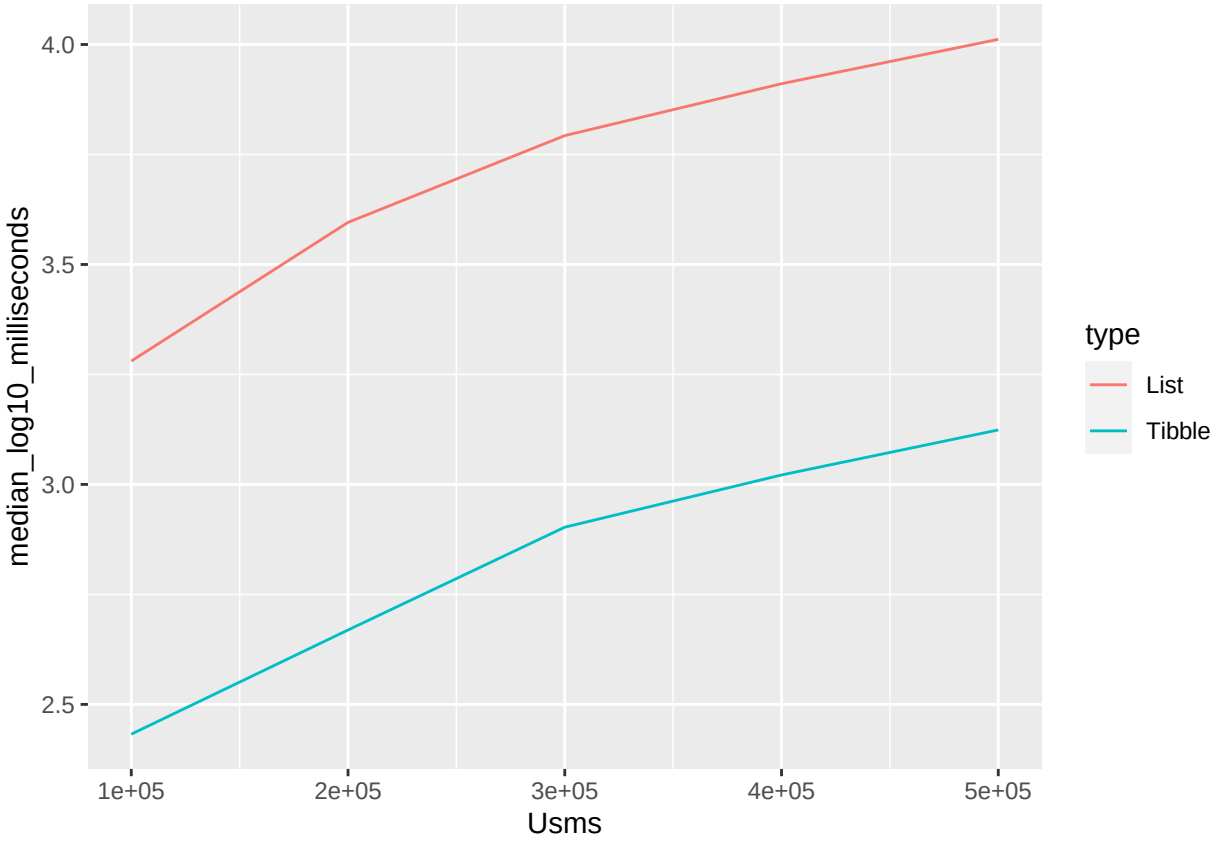
Multi-simulation's extraction test

For this test, we only have a single DoE, but we want to get the Usms names along with the values of the wanted variable for a specific date.

First setup : 1 DoE, Usms from 100K to 500K and 289 Dates Windows OS results :

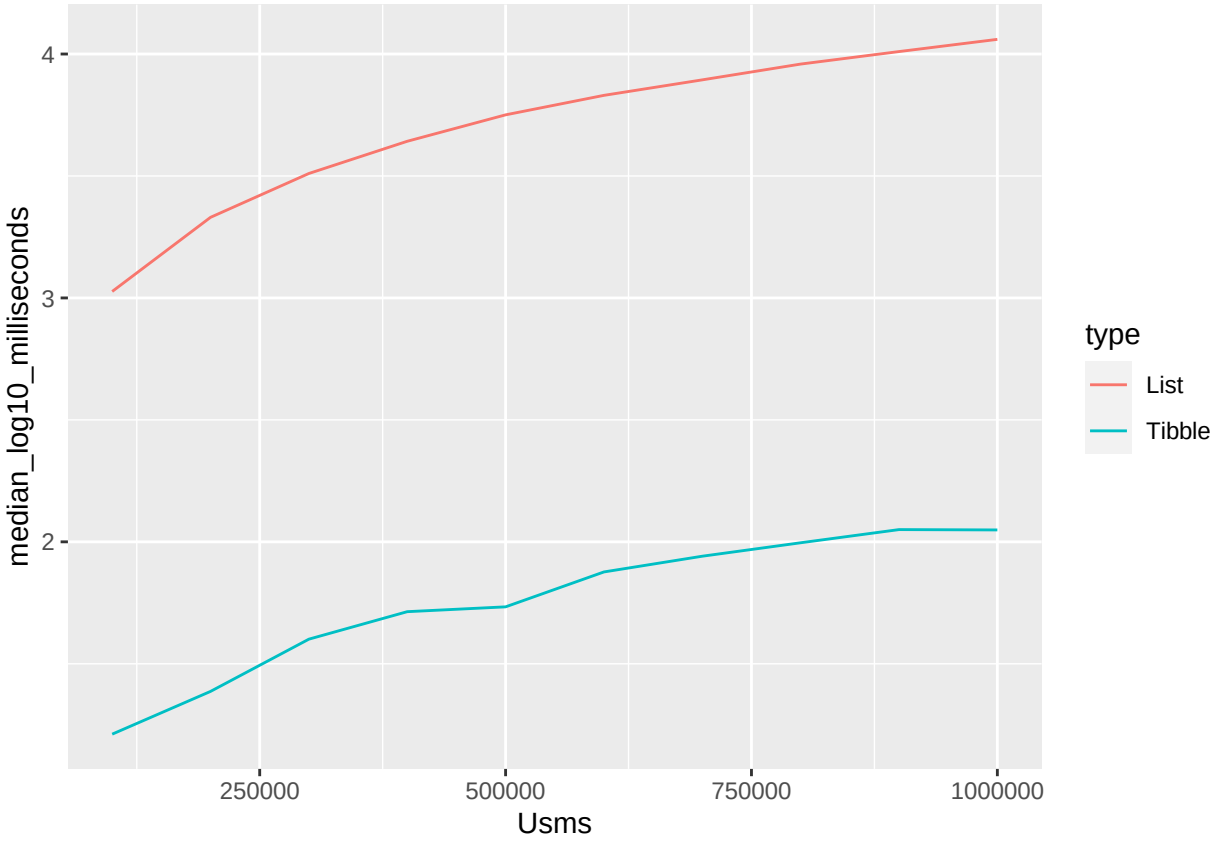


Linux OS results :

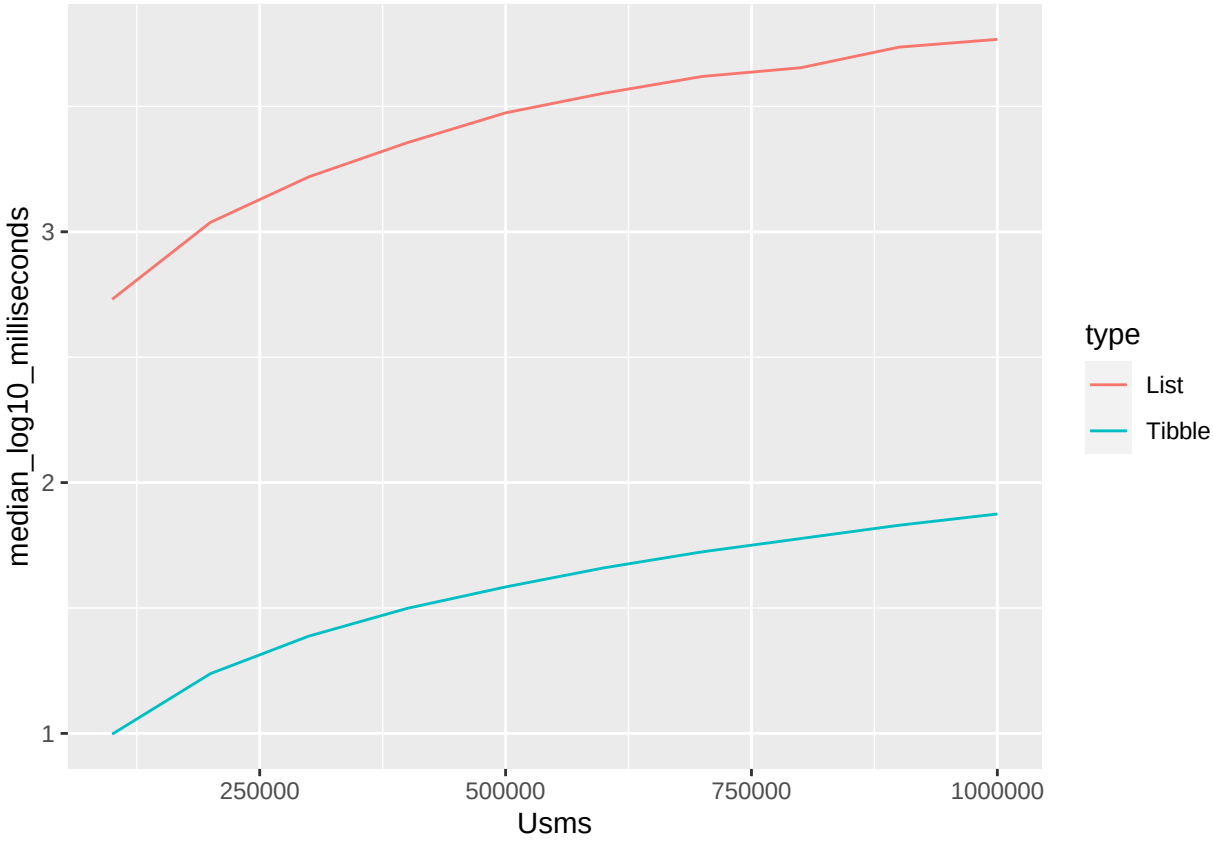


We can see that the Tibble is faster than the List in this first setup.

Second setup : 1 DoE, Usms from 100K to 1 Million and 5 Dates Windows OS results :



Linux OS results :



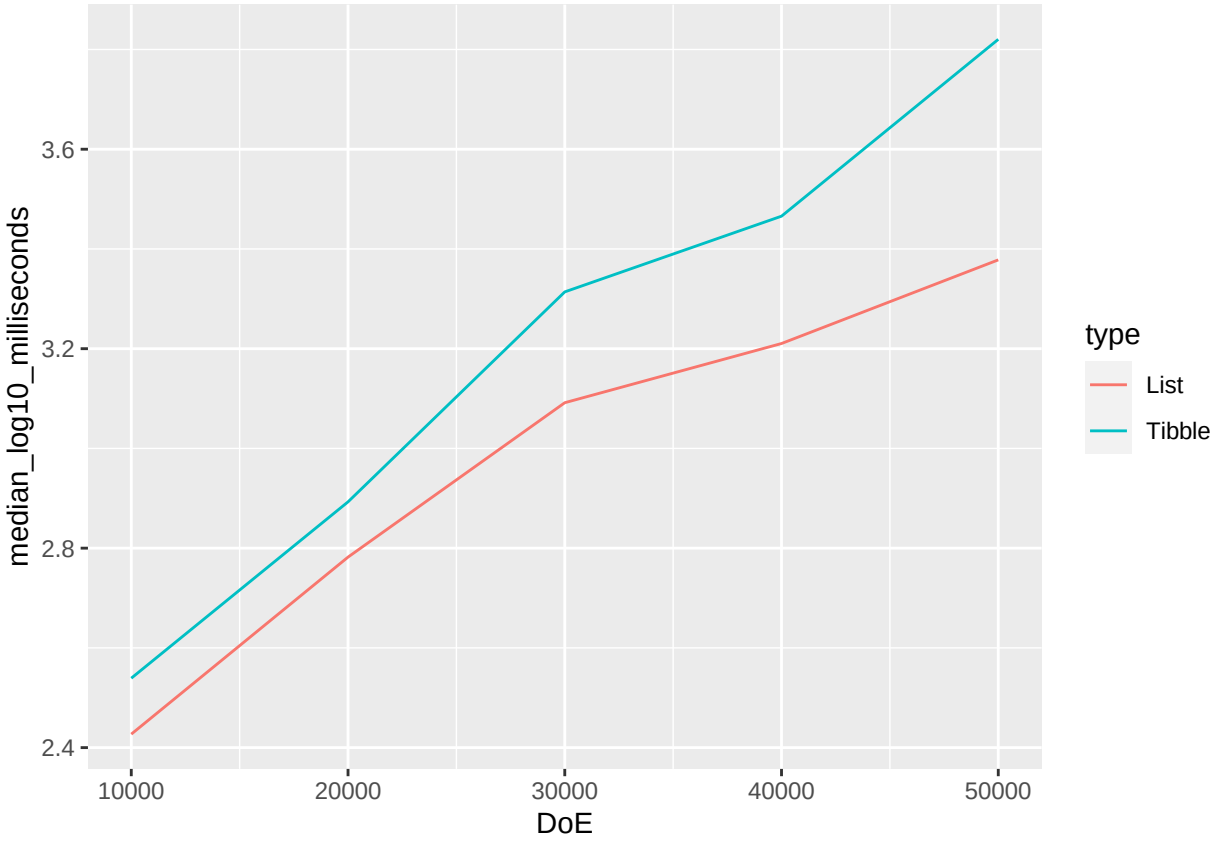
The results on this graph follow the results on the previous one. The Tibble storage method is more adapted to the multi-simulation use case than the List.

We can explain this by the fact that there are more operation done in the List function than in the Tibble function.

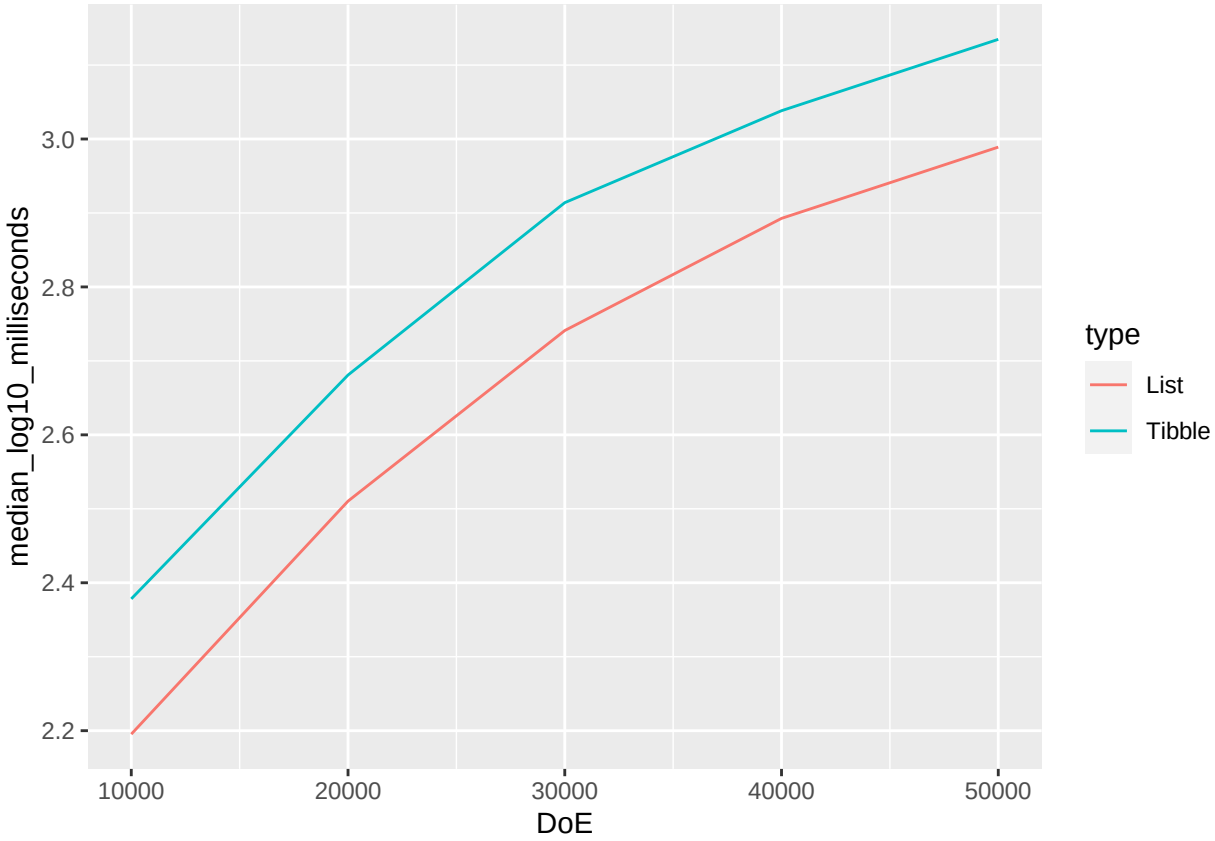
Analysis' extraction test

For this test, we get the DoE aong with the variable's values for the wanted variable for a specific Usm ans date.

First setup : DoE from 10K to 50K, 10 Usms and 289 Dates Windows OS results :

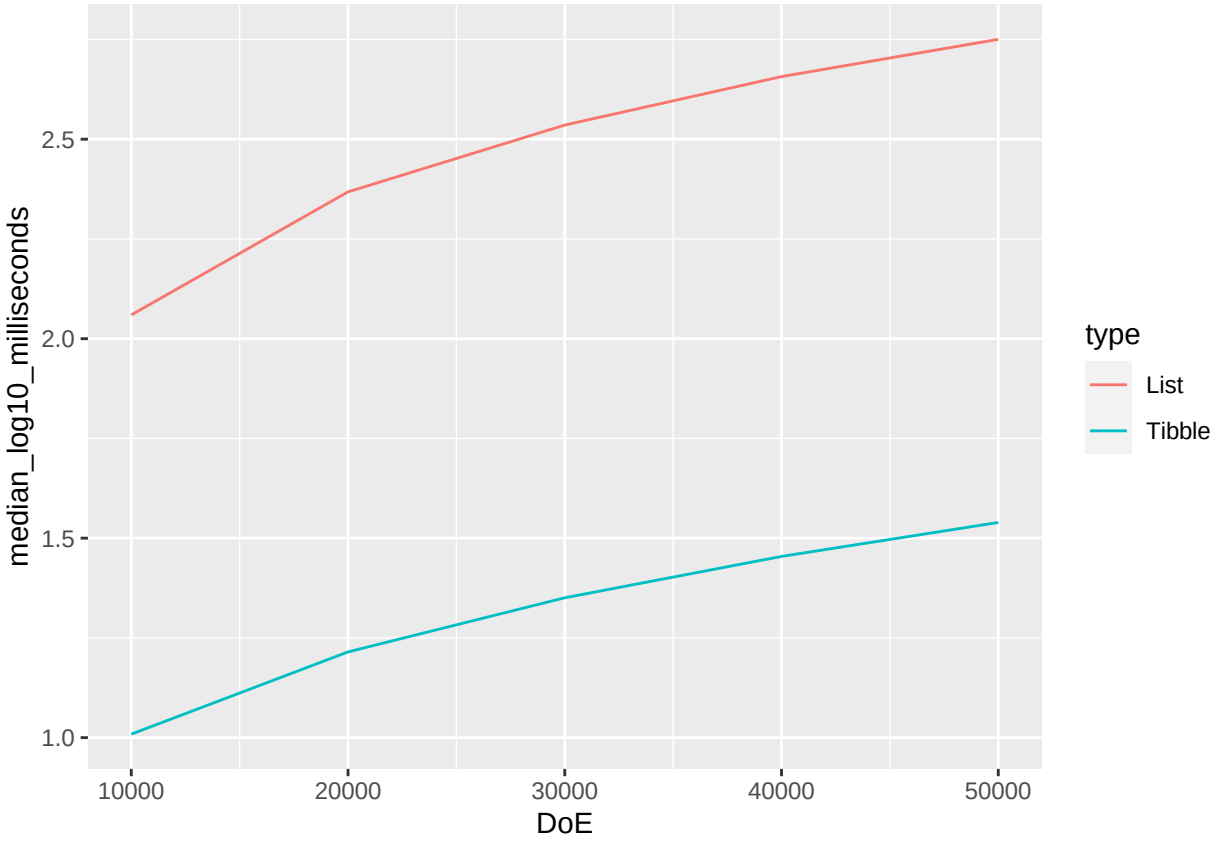


Linux OS results :

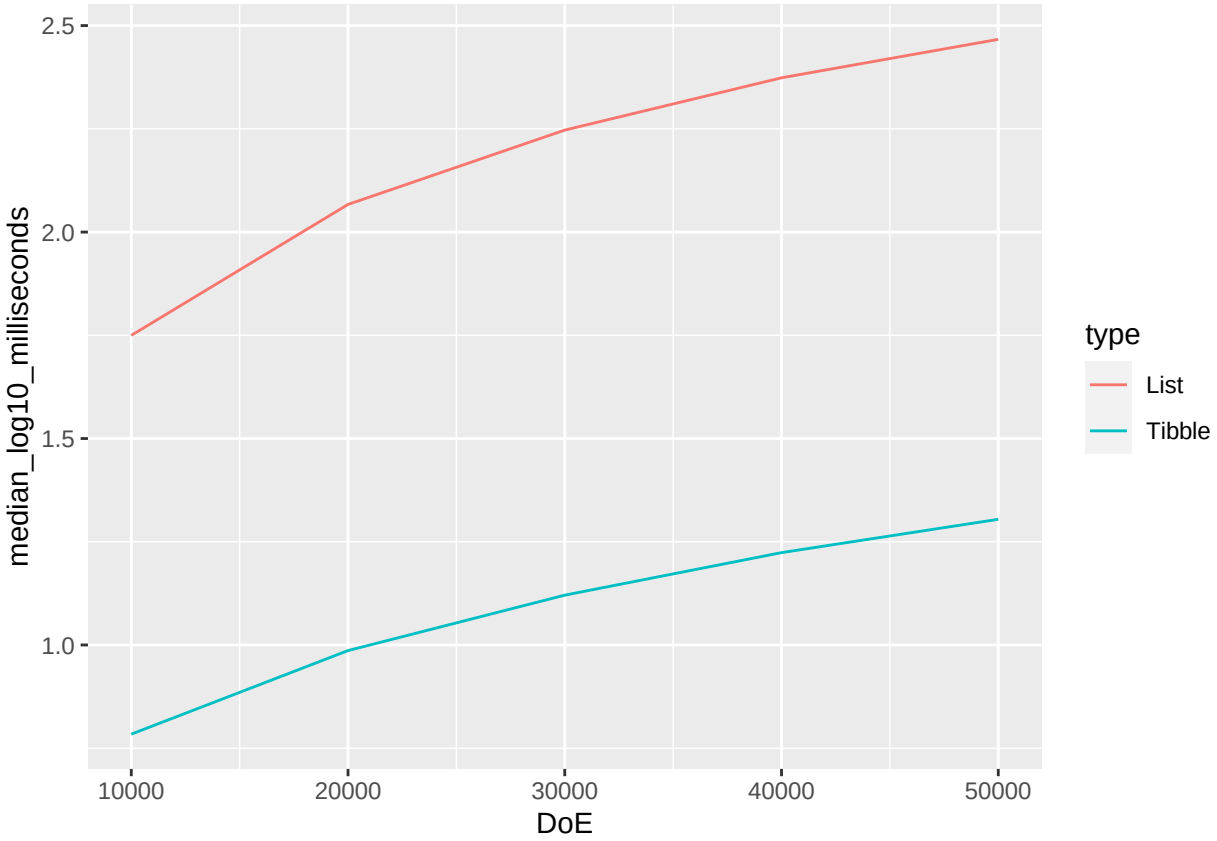


We can see that the List are faster than the Tibble on these tests.

Second setup : DoE from 10K to 50K, 10 Usms and 5 Dates Windows OS results :



Linux OS results :



Unlikely to the previous results, this time it's the Tibble that is faster than the List. We maybe can explain that on the way the dplyr::filter function has been coded. Indeed, in the List function, the wanted Usms are get first, then their variables and values are processed while in the Tibble function, each tibble's row is tested. And this process takes more time when they are 289 dates per Usm than when they are only 5. We can also conclude that it is faster to search information in a tibble using the dplyr::filter function than in a List.

Conclusion

First of all, the results we get using both OS are matching but we cannot say that they are concluant yet. The first reason is that we tested some use cases that are not very used which is the case for the optimization. This type of optimization is not the common one. The second reason is that we can see that there can be differences in results according to the number of dates you are using. Especially for the analysis case. The last reason is that we done these tests on an only one way to store the data which was by DoE by Usm but maybe there are others storage way which ca give better results or reversed results or both, that's why the results are not concluant yet but they give us some hints in the direction to follow.