MaxWiK

–**Max**ima **W**eighted **i**solation **K**ernel machine learning method–

2024/11/23

Contents

1 Input data

1.1 Data format

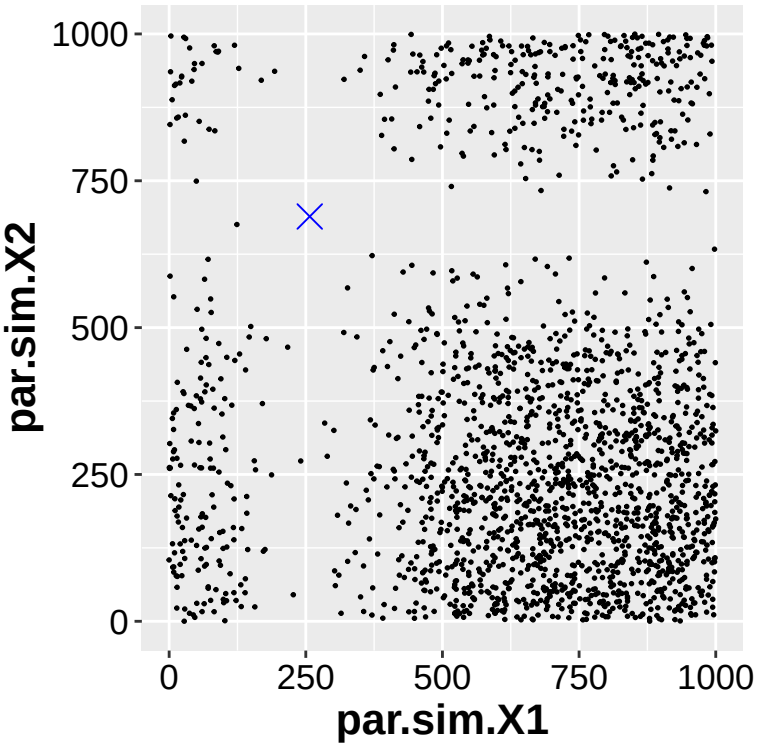
The data format of input should be numerical only in the form of data frame, for example:

```
MaxWiK::Data.2D$X[1:4, ]
#>   par.sim.X1 par.sim.X2
#> 1   512.4316 853.256380
#> 2   728.8298 388.103700
#> 3   636.5520   7.421515
#> 4    40.4254 306.872169
str(MaxWiK::Data.2D$X)
#> 'data.frame':    2000 obs. of  2 variables:
#>  $ par.sim.X1: num  512.4 728.8 636.6 40.4 513 ...
#>  $ par.sim.X2: num  853.26 388.1 7.42 306.87 164.56 ...
```

If you need use a matrix, please, transform it into the data frame first. If some columns have non-numerical type, please, transform them into numerical type using factors.

1.2 Specificity of the data

The method is mostly appropriate for specific data which has sparse regions of interest. The package dataset `Data.2D` has an example where the lack of data points at the region of the interest (blue cross):



1.3 Input for Approximate Bayesian Computation and metasampling algorithm

Inputs for Approximate Bayesian Computation (ABC) and metasampling algorithm have the same part. That includes parameters dataset, dataset of summary statistics of simulations, summary statistics of observation, and hyperparameters like t, ψ and Matrix_Voronoi. To reproduce the results of ABC, it is necessary to use the same hyperparameters including Matrix_Voronoi where the Voronoi sites for each tree are determined.

1.4 Input for sampling algorithm

Two functions are used for sampling: sampler_MaxWiK() and sampler_MaxWiK_parallel() which have the input part as for metasampling. All the arguments are described in the manual but the most important are as follow:

- model - the function of simulation;
- arg0 - list of constant arguments for the model function.

To run algorithm properly, a user has to check the simulation first like:

do.call(what = model, args = c(arg0, list(x = c(5, 7))))

where x is the variable.

1.5 Input for predictor

Input of the function MaxWiK.predictor() is the same as for metasampling algorithm, for details, please, see manual.

2 Output data

2.1 Output of MaxWiK algorithm

2.1.1 Voronoi matrix

The Voronoi matrix is a matrix of information about Voronoi sites for each tree in the isolation forest algorithm (rows - trees, columns - Voronoi sites IDs) for parameters data set:

```
cols = c(1:4, 9:16)
df = as.data.frame( MaxWiK::Data.2D$ABC$Matrix.Voronoi[ 1:4, cols ] )
names( df ) = cols
print( df )
#>     1   2   3   4    9   10   11   12   13   14   15   16
#> 1 191 226 258 445  767  965 1305 1585 1643 1701 1952 1972
#> 2 194 295 313 473 1335 1550 1570 1616 1663 1843 1939 1974
#> 3  10 147 213 255 1219 1571 1601 1622 1693 1703 1752 1983
#> 4  65  76 122 299 1261 1428 1465 1540 1687 1690 1692 1936
```

2.1.2 The kernel mean embedding

The kernel_mean_embedding is a kernel mean embedding of observation point in the form of the vector of IDs of the Voronoi sites in the Voronoi matrix for each tree:

```
MaxWiK::Data.2D$ABC$result.MaxWiK$kernel_mean_embedding
#> [1] 11 16  7  9 14  4  1 16
```

In this example, the kernel mean embedding has 8 IDs for 8 trees where each ID is a column number of Voronoi matrix, tree ID corresponds to the row of Voronoi matrix.

2.1.3 Similarity

The similarity vector is the similarity measured by using isolation kernel, and it shows how each data point similar to the observation one, for example for the first points:

```
MaxWiK::Data.2D$ABC$result.MaxWiK$similarity[1:20]
#>  [1] 0.625 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.375
#> [13] 0.000 0.000 0.000 0.000 0.000 0.250 0.000 0.875
```

2.1.4 Matrix_iKernel

The Matrix_iKernel is the matrix of Voronoi sites for each data point in the form of Hilbert representation, for example, for the first data points:

```
MaxWiK::Data.2D$ABC$result.MaxWiK$Matrix_iKernel[1:4,]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
#> [1,]   11    2    7    9   14    4    1    7
#> [2,]    6    5    4   14   10   11   16   13
#> [3,]   13    6    6    3   11    2    8    5
#> [4,]   12    9   10   10   12   12   10    8
```

2.1.5 Other data

Other data like hyperparameters, t, ψ, etc., can be found in the description of the related functions, or intuitively understandable.

2.2 Output of metasampling algorithm

The metasampling algorithm has several features, because it iteratively generates the data points in the space of parameters and selects the points with higher similarity. The result has the next issues:

- input.parameters is just all the input parameters of the function **meta_sampling()**;
- iteration is a number of iterations during metasampling;
- iKernelABC is the list of output from **get.MaxWiK()** function;
- par.best is the best parameter found at the end of metasampling;
- sim.best is the similarity corresponing to the par.best point;
- spiderweb is the list of the generated data points for each iteration, the name 'spiderweb' used because of algorithmic feature which generates data points closer and cloaser to the observation like spider makes a web.

2.3 Output of sampling algorithm

The functions sampler_MaxWiK() and sampler_MaxWiK_parallel() return the list of the next objects:

- results - the data frame of results of the sampling algorithm which includes the generated parameters, simulated output data, similarities (sim), mean squared error (mse), comments (com), and simulation IDs (sim_ID):

```
#> 'data.frame':    507 obs. of  8 variables:
#>  $ par.sim.X1 : num  474 479 469 484 464 ...
#>  $ par.sim.X2 : num  706 717 695 703 709 ...
#>  $ stat.sim.Y1: num  519 504 534 487 552 ...
#>  $ stat.sim.Y2: num  997 992 1000 998 996 ...
#>  $ mse        : num  231225 245789 217049 263022 201053 ...
#>  $ comm       : chr  "Network" "Network" "Network" "Network" ...
#>  $ sim        : num  1 1 1 1 1 1 1 1 1 1 ...
#>  $ sim_ID     : num  1 2 3 4 5 6 7 8 9 10 ...
```

- best - the data frame of the most accurate simulation:

```
#> 'data.frame':    1 obs. of  8 variables:
#>  $ par.sim.X1 : num 257
#>  $ par.sim.X2 : num 689
#>  $ stat.sim.Y1: num 1000
```

```
#>  $ stat.sim.Y2: num 1000
#>  $ mse        : num 6.96e-07
#>  $ comm       : chr "Network"
#>  $ sim        : num 1
#>  $ sim_ID     : num 502
```

- time - the time of sampling in seconds;
- n_simulations - the total number of simulations;
- number_of_iterations - the number of iterations, each iteration includes several simulations, and number of simulations in each iteration depends on hyperparameters.

2.4 Output of predictor

The output format of predictor is almost same as for metasampling algorithm, for details, please, see manual.

3 Templates

To use the templates, please, be kind to call the function:

```
MaxWiK::MaxWiK_templates(dir = tempdir())
```

The function MaxWiK_templates() copies all the templates to the working folder. There are 3 templates:

- MaxWiK.ABC.R - template to run approximate Bayesian computation based on MaxWiK algorithms including metasampling;
- MaxWiK.Sampling.R - template to run sampling generation and run a model with example of Gaussian function;
- MaxWiK.Predictor.R - template to get prediction based on metasampling procedure.

One can find the explanation of leveraging of the templates in this vignette.

4 Approximate Bayesian Computation

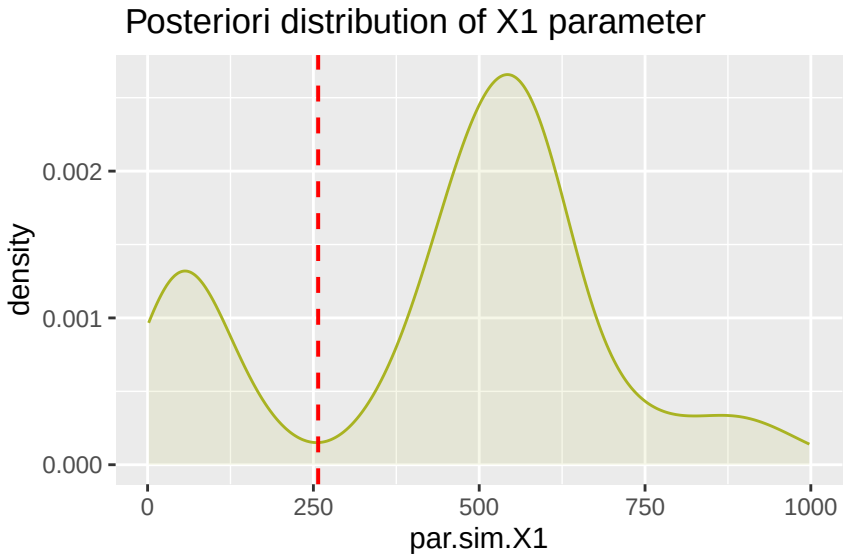
To get the approximate Bayesian computation for a given dataset, one can use function get.MaxWiK(), here, please, try 2D example:

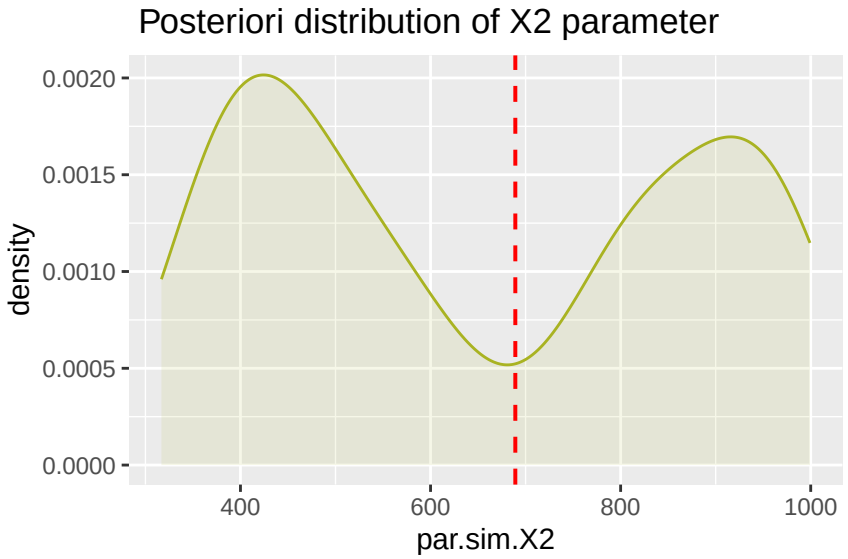
```
example.2D = MaxWiK::Data.2D
obs        = as.data.frame( t ( example.2D$observation$A ) )
Matrix.Voronoi  =  example.2D$ABC$Matrix.Voronoi
res.MaxWik = get.MaxWiK( psi = example.2D$ABC$hyperparameters$psi,
                         t   = example.2D$ABC$hyperparameters$t,
                         param    = example.2D$X,
                         stat.sim = example.2D$Y,
                         stat.obs =  obs,
                         talkative = TRUE,
                         check_pos_def = TRUE,
                         Matrix_Voronoi = Matrix.Voronoi )
```

To see the Posteriori a user can extract points with non-zero similarity:

```
w.sim = which(res.MaxWik$similarity > 0 )
posteriori.MaxWiK = Data.2D$X[ w.sim, ]
```

That looks like that (red dashed lines are true values of X1 and X2):




Posteriori distribution of X2 parameter
density
0.0020
0.0015
0.0010
0.0005
0.0000
400
600
800
1000
par.sim.X2

5 Metasampling

To improve the approximate Bayesian computation for a given dataset, one can use metasampling function meta_sampling() instead. So the usage for the same 2D example looks like:

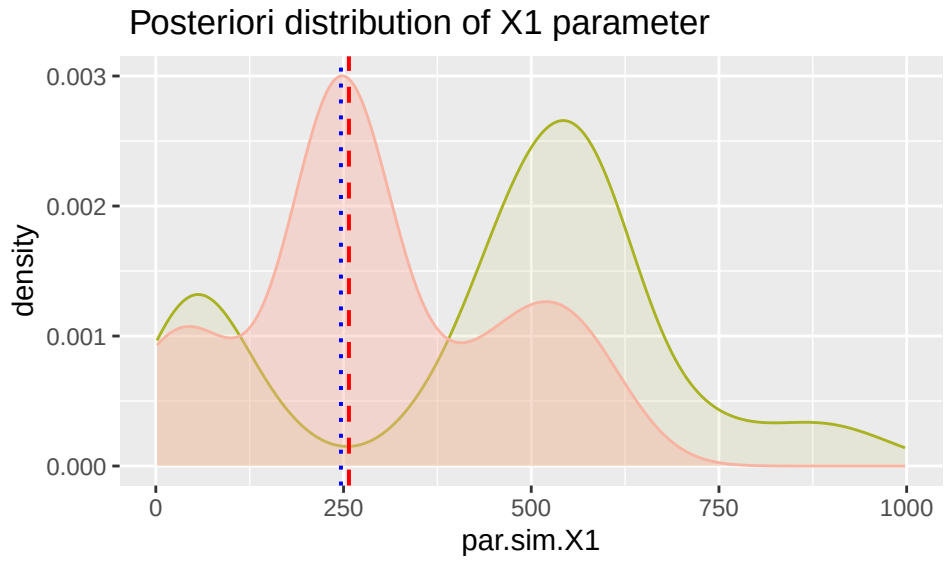
```
example.2D  = MaxWiK::Data.2D
obs         = as.data.frame( t ( example.2D$observation$A ) )
res.MaxWik  =  MaxWiK::Data.2D$ABC$result.MaxWiK
meta.sampling  =  meta_sampling( psi = example.2D$ABC$hyperparameters$psi,
                                 t   = example.2D$ABC$hyperparameters$t,
                                 param    = example.2D$X,
                                 stat.sim = example.2D$Y,
                                 stat.obs =  obs,
                                 talkative = TRUE,
                                 check_pos_def = TRUE,
                                 n_bullets = 42,
                                 n_best = 12,
                                 halfwidth = 0.5,
                                 epsilon = 0.001,
                                 rate = 0.2,
                                 max_iteration = 10,
                                 save_web = TRUE,
                                 use.iKernelABC = res.MaxWik )
```

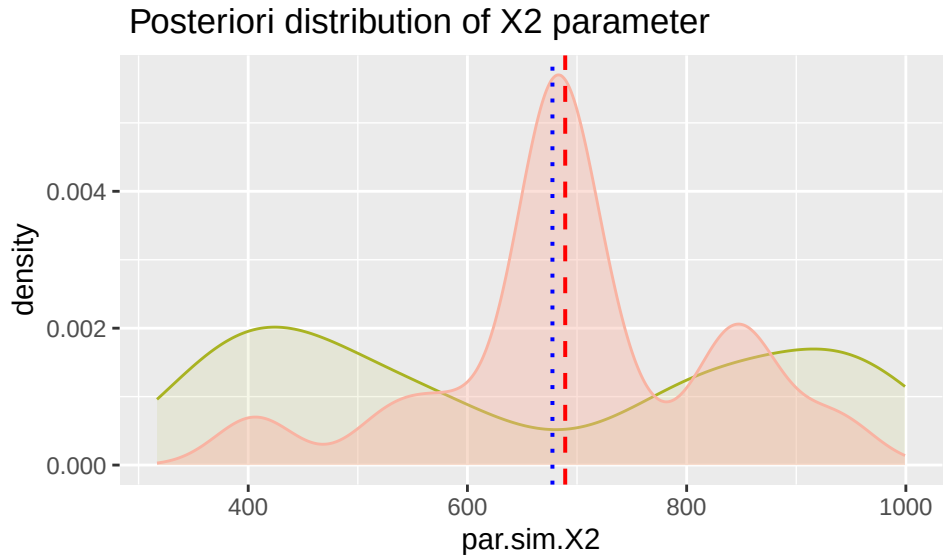
Please, pay attention that argument 'use.iKernelABC' can be skipped, by default it is set up to NULL and is generated for metasampling algorithm.

To get all the unique points from the 'spiderweb' list of the output data, one can run:

```
        network  = unique.data.frame( do.call(rbind.data.frame, meta.sampling$spiderweb ) )
```

There are plots for posteriori distribution of $X1$ and $X2$:





The red line is a true value, dotted is the best simulation, the red area is metasampling generated points (generated network), and the green area is the points taken from isolation kernel algorithm with similarity more than 0.

6 MaxWiK sampling

If a generating model function is known then the MaxWiK sampling algorithm can be run. The input is the same as for metasampling with the additional arguments for generating function:

- model - the function of simulation;
- arg0 - list of constant arguments for the model function.

To run algorithm properly, a user has to check the simulation first like:

```
do.call( what = model, args = c( arg0, list( x = c(5, 7) ) ) )
```

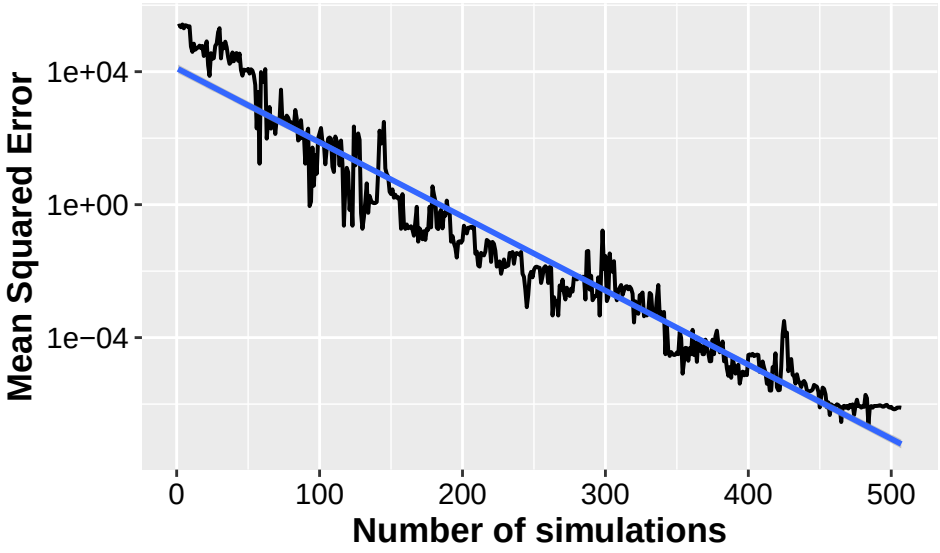
where x is the variable of the model function. Using the same 2D example the sampling can be run as follow:

```
smpl.2D  = MaxWiK::Data.2D$sampling
stat.sim = MaxWiK::Data.2D$Y
par.sim  = MaxWiK::Data.2D$X
sampling.res = sampler_MaxWiK( stat.obs =  smpl.2D$stat.obs,
                               stat.sim =  stat.sim,
                               par.sim  =  par.sim,
                               model    =  smpl.2D$model_function,
                               arg0     =  smpl.2D$model_par,
                               size     =  1600,
                               psi_t    =  smpl.2D$psi_t,
                               epsilon  =  1E-10,
                               check_err  =  FALSE,
                               nmax     =  60,
                               include_top  =  TRUE,
                               slowly       =  TRUE,
                               rate         =  0.05,
                               n_simulation_stop = 1000,
                               include_web_rings = F,
                               number_of_nodes_in_ring = 1  )
```

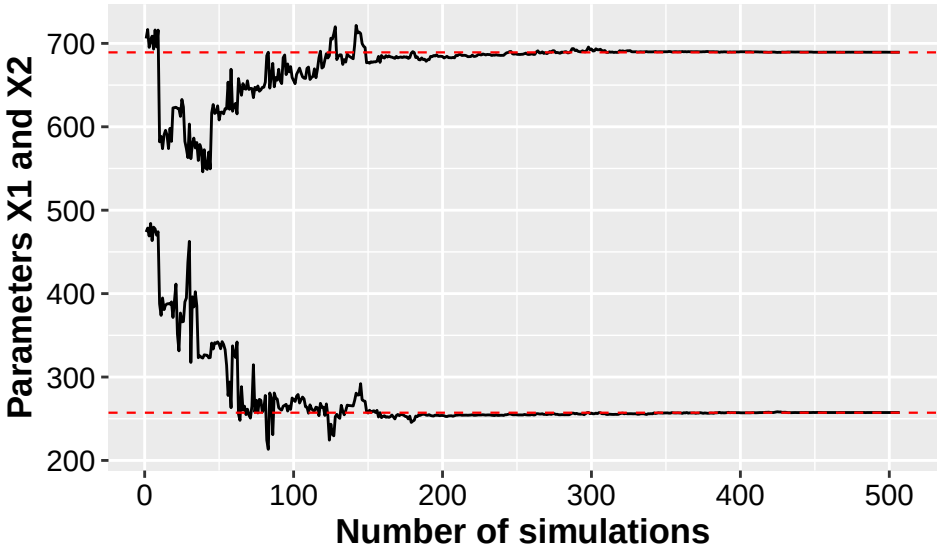
To see the mean squared error evolution and seeking process of the parameters, please, just use data from the simulation:

```
MSE  =  data.frame( sim_ID = sampling.res$results$sim_ID,
                    MSE    = sampling.res$results$mse )
X12  =  data.frame( sim_ID  = sampling.res$results$sim_ID,
                    X1 = sampling.res$results$par.sim.X1,
                    X2 = sampling.res$results$par.sim.X2 )
```

So, using results data frame, one can plot MSE of the sampling:



Also, the data generation can be visualized like for 2D example:



Here the red dashed lines are truth values of the observation.

7 Predictor

MaxWiK metasampling algorithm can be used for prediction. For this purpose one can run function MaxWiK.predictor(), here is the 2D demonstration:

```
pred.input  =  MaxWiK::Data.2D$predictor$result$input.parameters
stat.sim    =  MaxWiK::Data.2D$Y
par.sim     =  MaxWiK::Data.2D$X
new.param   =  as.data.frame( t( MaxWiK::Data.2D$observation$x0 ) )
iKernelABC  =  MaxWiK::Data.2D$predictor$result$iKernelABC
predictor   =  MaxWiK.predictor( psi = pred.input$psi,
                                 t   = pred.input$t,
                                 param     = par.sim,
                                 stat.sim  = stat.sim,
                                 new.param = new.param,
                                 talkative = FALSE,
                                 check_pos_def = FALSE ,
                                 n_bullets  = 42,
                                 n_best     = 12,
                                 halfwidth  = 0.5,
                                 epsilon    = 0.001,
                                 rate       = 0.2,
                                 max_iteration = 10,
                                 save_web   = TRUE,
                                 use.iKernelABC = iKernelABC
)
```

Here the iKernelABC is the results of MaxWiK ABC algorithm. So, the most probable value of the function is

```
predictor$prediction.best
```

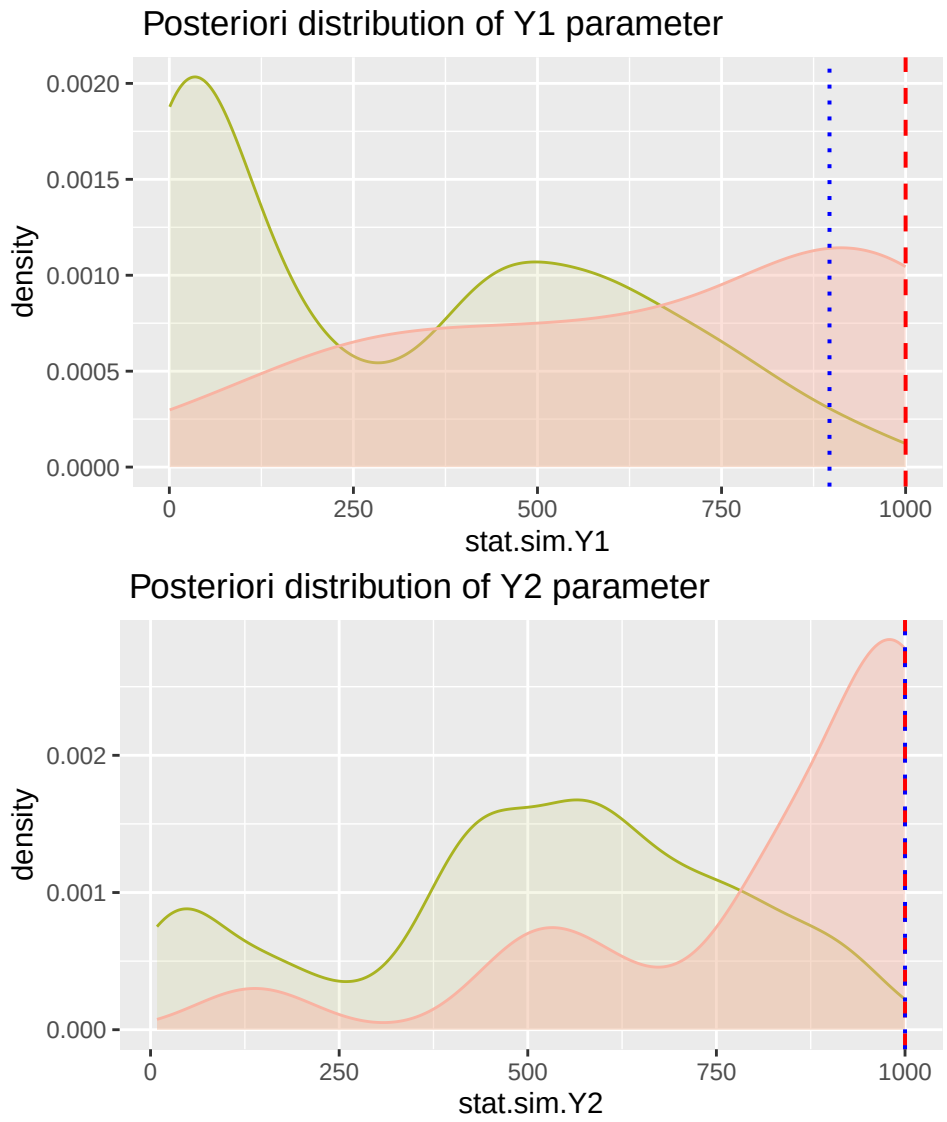
To see the distribution of the points generated by metasampling one can get a metasampling 'network':

```
pred.network  = unique.data.frame( do.call(rbind.data.frame, predictor$spiderweb ) )
```

In principle, MaxWiK algorithm is able to produce metasampling points out of the reasonble range. So, please, pay attention that results should be restricted by possible range of values for each dimension, for example:

```
pred.network  = apply_range( diapason = c(0,1000), input.data = pred.network )
predictor$prediction.best = apply_range( diapason = c(0,1000),
                                input.data = predictor$prediction.best )
```

There are plots for posteriori distribution of $Y1$ and $Y2$ after the range application:



Here the red line is a true value, blue one is the best prediction, the green area is the posteriori based on the MaxWiK ABC algorithm, the red area is metasampling.

8 References

For publication, please, be kind to use next references related to MaxWiK software:

- Iurii S. Nagornov, Approximate Bayesian Computation Based on Maxima Weighted Isolation Kernel Mapping, arXiv.2201.12745, 2022
- Iurii S. Nagornov, Overfitting Problem in the Approximate Bayesian Computation Method Based on Maxima Weighted Isolation Kernel, The 36th Annual Conference of the Japanese Society for Artificial Intelligence, 2S5-IS-2c-05, 2022
- Package MaxWiK: https://github.com/tugHall/MaxWiK

9 Collaboration

Leveraging of MaxWiK package is free for any purpose. Adjusting hyperparameters for the particular task and certain datasets is the issue of collaboration. To get hyperparameters we use our Know-How algorithm which cannot be distributed freely. In each specific case this is being discussed, but in general the rules are as follow:

- for **commercial purpose** the adjusting of hyperparameters including Matrix of Voronoi sites is under commercial offer. Copyrights of hyperparameters' dataset belong to customers.
- for **research purpose** the adjusting of hyperparameters including Matrix of Voronoi sites is the cooperation issue. Copyrights of hyperparameters' dataset belong to us.