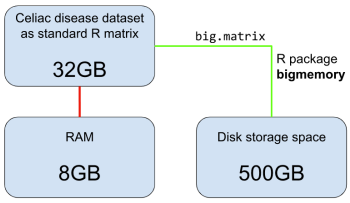

The main purpose of the $bigmemory$ package is to allow us to work with big data, potentially more than the RAM available in our devices.

Published by Michael J. Kane, John W. Emerson, Peter Haverty, and Charles Determan Jr. in 2012, while its latest version the 4.5.36 appeared in 2019 .

Alternatively, you can read only part of the X matrix, check all the variables in that part, and then read another part. We can also create, store, access and manipulate massive arrays.

Suite of packages Many specialized, with more advanced functionalities, packages accompany the $bigmemory$ package such as $biganalytics$, $synchronicity$, $bigalgebra$, and $bigtabulate$.

Many functions support the $big.matrix$ objects with the most important are $big.matrix()$, $mwhich$(similar to R's which), $read.big.matrix$(create a binary file-backing and a descreption file for future use, with a C++ pointer to the matrix on the memory), $write.big.matrix$(write it in a file), deepcopy(duplicate the matrix with different pointer). For **shared-memory** functionality: $shared.big.matrix()$, attach.big.matrix(get the matrix from the description file which is way less memory time consumption), $shared.deepcopy()$, $describe()$. For **statistical analysis** methods: $biglm()$, $summary()$, $biglm.big.matrix()$ or $bigglm$, $kmeans.big.matrix()$.

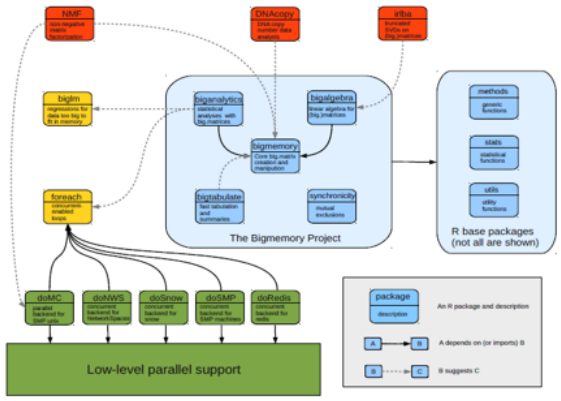

Figure 1: The Bigmemory Suite of Packages

The linear model for the $big.matrix$ object is based on considering the linear model n > p: $y = X\beta + \epsilon$ so the least square estimation is $\hat{\beta} = (X^T X)^{-1} X^T y$

For tall data, the R's basic $lm.fit$ use $O(np+p^2)$ of the memory while the $biglm$ uses only $O(p^2)$ so this is where the lead is in favour of the latter, as it tries to compute the decomposition of $X = QR$ and $Q^T y$ which leads us to β

$$R\beta = Q^T y$$

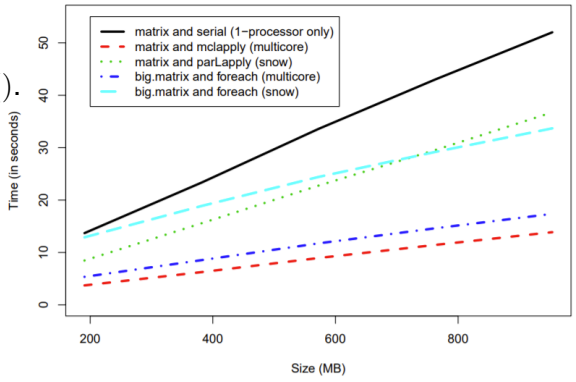

Figure 2: The benchmark results for varying simulation sizes using four processor core

by calling update, the fitted object can be updated with more data.

For **parallel computing**, we can use $foreach$ which is based on the computation by chunk of the $big.matrix$ objects X and Y in this way,

$$(X^T Y) = \sum_{g=1}^{G} X_{(g)}^T Y_{(g)}$$