# High dimensional surrogacy: computational aspects of an upscaled analysis

Rudradev Sengupta[a,c], Nolen Joy Perualila[c], Ziv Shkedy[a,b], Przemyslaw Biecek[d], Geert Molenberghs[a,b], Luc Bijnens[a,c]

[a]Center for Statistics (CenStat), Hasselt University, Hasselt, Belgium

[b]Interuniversity Institute for Biostatistics and statistical Bioinformatics (I-BioStat), Belgium

[c]Janssen Pharmaceutical companies of Johnson and Johnson, Beerse, Belgium

[d]Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland

**ARTICLE HISTORY**

Compiled June 25, 2019

## Supplementary Appendix

CONTACT Rudradev Sengupta. Email: rudradevsengupta@gmail.com

## 1. Introduction

This supplementary appendix contains additional materials related to the article. In Section 2, we present the "prologue" and "epilogue" framework that allows to upscale the analysis to larger datasets. In Section 3, we review the main steps in the `R` implementation of the master-slave framework discussed in the paper.

## 2. The Prologue-Epilogue Framework

Figure SA1 illustrates how to run the joint model on a relatively large dataset that can lead to a high average data-loading time. In this case, as we mentioned in Section 6 of the manuscript, we recommend to use the "prologue-epilogue" setting within the worker framework.

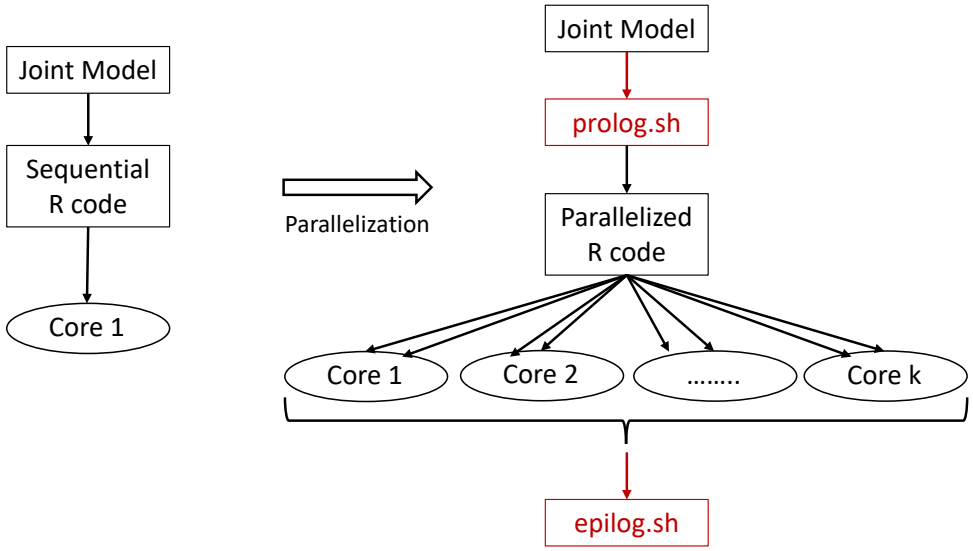


Figure SA1.: Worker framework with prologue and epilogue.

## 3. Implementation of the Worker Framework in R

In this section, we give an overview of the `R` codes for the implementation of the data analysis presented in the paper. Detailed code is available upon request.

### *3.1. For a Single Machine*

For a single fingerprint feature, one can use the `fitjm()`, from `IntegratedJM R` package, which uses the sequential `for loop` to fit the model for all the genes.

```
library(IntegratedJM)
jmRes <- fitJM(dat=X, responseVector=y, covariate=z,
               methodMultTest='fdr')
```

As mentioned in the manuscript in Section 4.2, we can use other `R` functions to do parallelization. To use `foreach()` the source code of the `R` function `fitJM()` was customized by replacing the `for loop` by a `foreach loop` and a parallel backend was registered by using the `makeCluster()`.

```
library(doParallel)
cl <- makeCluster(n) #n is the number of available cores
registerDoParallel(cl)
jmRes <- fitJMfe(dat=X, responseVector=y, covariate=z,
                 methodMultTest='fdr')
```

For the `clusterApply()` and `clusterApplyLB()`, the function `fitJM()` was modified to fit the joint model for a single gene and was renamed as `fitOneJM()`.

```
## J is the total number of genes
jmRes <- clusterApply(cl, 1:J, fun=fitOneJM, dat=X,
                      responseVector=y, covariate=z,
                      methodMultTest='fdr' )

jmRes <- clusterApplyLB(cl, 1:J, fun=fitOneJM, dat=X,
                        responseVector=y, covariate=z,
                        methodMultTest='fdr' )
```

### *3.2. For VSC Cluster*

It is possible to use the similar functions as described above in a single node of the VSC Cluster. In this section, we provide an overview of the code for the worker framework. The code presented below was used for the setting where 190 genes were allocated to a jobitem in the cluster (see Section 4.3 in the manuscript). Here as well, the function `fitJM()` was modified and renamed as `fitJMworker()`.

```
## to get the values of the parallelization parameter
## from the .csv file
args <- commandArgs(TRUE)
irun <- as.integer(args[1])

## creating chunks of 190 genes
Genes = 190
begin = (irun-1)*Genes + 1
end = irun*Genes
if(end<3595){
      X <- genematrix[begin:end,]
} else {
      X <- genematrix[begin:3595,]
}

## fitting the joint model
jmRes <- fitJMworker(dat=X, responseVector=y, covariate=z,
                     methodMultTest='fdr')
```