**BACKGROUND**

The *episcan* implementation here discussed is based on the (T et al., 2011), (Kam-Thong et al., 2010) papers. The implementation is largely based on the <https://cran.r-project.org/web/packages/episcan/index.html> package. The first step is totally based on the *epiblaster1geno* function in that library.

As stated in the outline of the library, *epiblaster1geno* can handle both **continuous** and **binary phenotype**, with the adequate flag in the methods.

A preliminar check was done to be sure about the results of the method and the markdown in a doc form can be retrieved in ../ Report\_episcan.docx

**CODE**

The code is based on two main bash files that iterates on every file in the data folder and launch the corresponding R implementations of the 1st and 2nd step case/control or with a continuous phenotype.

Those files name are

* iterator\_on\_files.sh
  + The file that parse the parameters and create an iteration loop with all the .txt files in the data folder
* script\_with\_parameter.sh
  + For each data input, launch the 2 *R scripts* with the corresponding flag for continuous/binary /case\_control, hence performs the prioritization and return the final file.

The implementations are in the R files. For the *case\_control* the files are

* [Episcan\_experiment\_case\_control.R](https://github.com/FedericoMelograna/Epiblaster_implementation/blob/main/Code/Episcan_experiment_case_control.R)
* [Episcan\_experiment\_2nd\_step\_case\_control.R](https://github.com/FedericoMelograna/Epiblaster_implementation/blob/main/Code/Episcan_experiment_2nd_step_case_control.R) 2nd step

While the respective for the *continuous* case are

* [Episcan\_experiment\_continuous.R](https://github.com/FedericoMelograna/Epiblaster_implementation/blob/main/Code/Episcan_experiment_continuous.R) //1st step
* [Episcan\_experiment\_2nd\_step\_continuous.R](https://github.com/FedericoMelograna/Epiblaster_implementation/blob/main/Code/Episcan_experiment_2nd_step_continuous.R) //2nd step

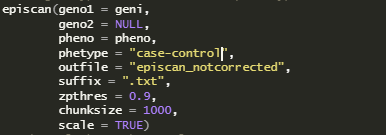
Focal functions in the scripts are

Immagine che contiene testo

Descrizione generata automaticamente

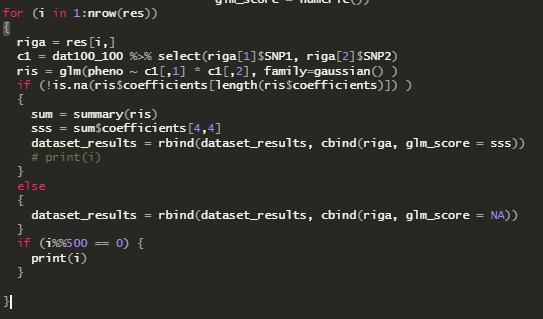
*1st step* for a quantitative outcome

For the binary case the function is

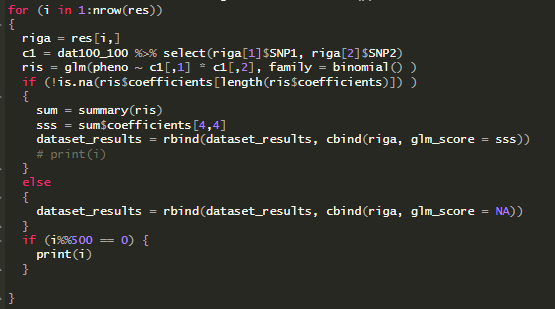


.

While for the *2nd step* the core function implementations are



Taking into account the coefficients of an interaction [ *the beta value*] in a LM regression for the continuous case



A ***GLM*** with binomial family for the binary case.

While



And 

Are the core function to *truncate* and *order* the results in the bash iterator.

For the truncating an adaptive behaviour is used in which a ***low threshold*** of 2.5 is applied if the interaction passing the first selection are not too many and ( > 500 kb ), otherwise a *stricter* threshold is used, with 3.5 and if the dimension is still > 500 kb, the top threshold of 4.5 is used.

**DATA SHAPE**

N.b. the data should be in a “ “ *space separed* file in which the outcome is in a column named “Class” and each other column is a SNP with the column name the SNP name

**RESULTS**

The results will be stored into a created folder with the same name as the input dataset [e.g. IBD\_DATA/]

And all preliminary and final results will be stored there.

The main output is:

episcan\_GLM\_second\_SORTED



Difference from *episcan\_GLM\_second\_SORTED* and *GLM\_second\_step\_NA\_elim* are only vestigial and are to be consider as interchangeable.

The first columns *SNP1* and *SNP2* are the name of the SNPs involved, while the *Z\_score* is the statistic, *ZP* is the p-value found in the 1st step, that is based on an approximation of the real likelihood function, while the *glm\_score* is the real p-value of the beta when adding the interaction of the predictors in the 2nd step.

Those p-values are ***not*** corrected for multiple testing.

**IMPLEMENTATION EXAMPLE**

**cd** /mnt/c/Users/fmelo/Desktop/Backup\_Federico/Work/Epistasis/Episcan\_epiblaster/Pipeline\_iterative/Code\_v2\_enhanced

**./iterator\_on\_files.sh** /mnt/c/Users/fmelo/Documents/GitHub/epistasis-simulation/Gametes/Data\_li\_1\_EDM-2/ /mnt/c/Users/fmelo/Desktop/Backup\_Federico/Work/Epistasis/Episcan\_epiblaster/Pipeline\_iterative/Gametes\_results/ /mnt/c/Users/fmelo/Desktop/Backup\_Federico/Work/Epistasis/Episcan\_epiblaster/Pipeline\_iterative/Code\_v2\_enhanced/ quantitative

Or, in a more generalizable way

**cd** Documents/GitHub/epiblaster/

**filepath**=`pwd`

**# mkdir** -p Example\_result # the code can also automatically create the repo!

**Code/iterator\_on\_files.sh** $filepath/Example\_data/ $filepath/Example\_result/ $filepath/Code case\_control

The last flag “case\_control” is a binary flag that take into account if the **phenotype** is *binary (*case\_control ) or continuous (continuous*).*

We refer to the README to further information on how to set a successful one-liner implementation

**PACKAGE NEEDED**

The only needed packages are

library(episcan)

library(dplyr)

That can be easily installed from the CRAN.

The implementation is based on a UNIX system, so it naturally adapts to the server, but need **git bash** or similar to run on a WINDOWS machine. It has been tested on both a windows machine with wsl.exe ( Ubuntu 20.04 LTS (GNU/Linux 4.4.0-19041-Microsoft x86\_64) ) and the server.

Kam-Thong, T., Czamara, D., Tsuda, K., Borgwardt, K., Lewis, C. M., Erhardt-Lehmann, A., Hemmer, B., Rieckmann, P., Daake, M., Weber, F., Wolf, C., Ziegler, A., Pütz, B., Holsboer, F., Schölkopf, B., & Müller-Myhsok, B. (2010). EPIBLASTER-fast exhaustive two-locus epistasis detection strategy using graphical processing units. *European Journal of Human Genetics 2011 19:4*, *19*(4), 465–471. https://doi.org/10.1038/ejhg.2010.196

T, K.-T., D, C., K, T., K, B., CM, L., A, E.-L., B, H., P, R., M, D., F, W., C, W., A, Z., B, P., F, H., B, S., & B, M.-M. (2011). EPIBLASTER-fast exhaustive two-locus epistasis detection strategy using graphical processing units. *European Journal of Human Genetics : EJHG*, *19*(4), 465–471. https://doi.org/10.1038/EJHG.2010.196