MAGAR: Methylation-Aware Genotype Association in R

Michael Scherer

September 14, 2020

Introduction

This vignette describes the package "Methylation-Aware Genotype Association with R* (*MAGAR*) available from GitHub. *MAGAR* uses DNA methylation data obtained using the Illumina BeadArrays, and genotyping data from Illumina genotyping microarrays or whole genome sequencing to compute methylation quantitative trait loci (methQTL). The package provides mutliple flavors of linear modeling strategies to compute *methQTL* as statistically significant interactions between single nucleotide polymorphisms (SNPs) and changes in the DNA methylation state of individual CpGs. DNA methylation values at single CpGs are first summarized into correlation blocks, and a representative of this correlation block (tag-CpG) is used in the methQTL calling.

Installation

The package relies on some dependencies that are exclusively available from Bioconductor and not from CRAN, most notably *RnBeads*. After installing the dependencies, *MAGAR* can be installed from GitHub, after installing the *devtools* package.

Required external software tools

MAGAR depends on a funtional installation of *PLINK* for handling genotyping data. Follow the installation instructions available here and specify the path to the installation for the package:

```
qtlSetOption(plink.path="path_to_plink")
```

```
## Error in qtlSetOption(plink.path = "path_to_plink"): could not find function "qtlSetOptic
```

Additionally, depending on the type of the analysis, more external software tools are required. More specifically, if you want to perform imputation *bgzip* and *tabix* from the *htslib* package are needed, as well as *vcftools* for sorting VCF files.

Lastly, if methQTL calling is to be performed through *fastQTL*, the software tool needs to be installed and the location of the executable specified to *MAGAR*.

```
qtlSetOption(
    bgzip.path="path_to_bgzip",
    tabix.path="path_to_tabix",
    vctools.path="path_to_vcftools_folder",
    fast.qtl.path="path_to_fastQTL",
)
```

Input data

MAGAR uses two types of data as input: DNA methylation data obtained using the Illumina Infinium BeadArrays or bisulfite sequencing and genotyping data obtained using genotyping microarrays or whole genome sequencing.

DNA methylation data (microarrays)

MAGAR utilizes the widely used *RnBeads* software package for DNA methylation data import. Thus, *MAGAR* supports the various input options available in *RnBeads*, including a direct download from the Gene Expression Omnibus (GEO), IDAT, and BED files. For further options, we refer to the RnBeads vignette and documentation. In addition to the raw methylation data, a sample annotation sheet specifying the samples to be analyzed needs to be provided. The sheet contains a line for each sample and looks as follows:

```
SampleID,age,sex,barcode
Sample_1,14,f,209054857842_R01C01
Sample_2,42,f,209054857842_R02C01
Sample_3,45,m,209054857842_R03C01
```

For further details on the import process, we refer to the RnBeads vignette. Most importantly, analysis options need to be specified for the import and preprocessing modules of *RnBeads*. *MAGAR* provides a default setting, which is available in *extdata/rnbeads_options.xml*. You can use this file as a template for your own setting and then specify it to the package:

```
opts <- rnb.xml2options(system.file("extdata/rnbeads_options.xml",package="methQTL"))
rnb.options(identifiers.column="geo_accession",
    import.idat.platform="probes450")
xml.fi <- file.path(getwd(),"rnbeads_options.xml")
cat(rnb.options2xml(),file=xml.fi)
qtlSetOption(rnbeads.options = xml.fi)
```

To redefine the correlation blocks, we allow for including additional information such as genome-wide segmentation of the methylation landscape (see option `use.segmentation` and function `qtlRunSegmentation`), and also functional annotation according to the Ensembl regulatory build [@Zerbino2015].

Genotyping data

PLINK files

MAGAR supports data that has already been processed by *PLINK* and that is available either in the form of binary *.bed*, *.bim* and *.fam* files, as *.ped* and *.map*, as variant calling files (*.vcf*), or as imputed files in the dosage format (*.dos*). For further processing, we use the command line tool *PLINK*, which is shipped together with *MAGAR*. However, this installation is only valid for Linux systems. For Windows and MacOS users, please install the *PLINK* tool from here and specify it using the option `plink.path`. The sample identifier specified earlier also needs to match the sample IDs of the genotype calls.

IDAT files

MAGAR also supports raw IDAT files and uses the *CRLMM* R-package, together with PLINK to perform genotype calling and data import. The package requires a single sample annotation sheet in the format described in the DNA methylation data section. In addition to the column names specified above, a column named *GenoSentrixPosition* has to be added, which specifies the IDAT file IDs.

```
SampleID,age,sex,barcode,GenoSentrixPosition
Sample_1,14,f,209054857842_R01C01,9701756058_R05C01
Sample_2,42,f,209054857842_R02C01,9701756058_R07C01
Sample_3,45,m,209054857842_R03C01,9742011016_R04C01
```

Imputation

Since Illumina SNP BeadArray data is typically imputed before further analysis, the package integrates a imputation functionality through the Michigan Imputation Server. Using the option setting described below, the package will automatically submit imputation jobs to the server and process the resulting files. In order to be able to perform computation on the server, an account is required. After the account is created, one has to request an API token in the user settings and specify it to *MAGAR* using the option `imputation.user.token`. During the imputation process, the package will stall for a while and wait for the job to finish. After the job is completed, the package will prompt for entering the password send via e-mail to the user account. The imputation process has to be split according to chromosomes, which is why multiple e-mails will be send to the account, and the imputation process can take up to several days. However, after imputation, the imputed data will be available as PLINK files, such that the imputation has to be performed only once. For preprocessing the data for upload to the imputation server, the package requires the bgzip and tabix tools from the htslib package. Also see further options to configure the imputation jobs at the Michigan Imputation Server documentation:

```
qtlSetOption(
  impute.geno.data=TRUE,
  imputation.reference.panel="apps@hrc-r1.1",
  imputation.phasing.method="shapeit",
  imputation.population="eur"
)
```

Perform data import

The `doImport` function requires the paths to the respective genotyping and DNA methylation data, as well as a sample annotation sheet as discussed earlier. In this vignette, we will describe the import of DNA methylation data in *IDAT* format and genotyping data as *PLINK* files. First, you'll have to specify the paths to the corresponding *IDAT* and *plink* files. Additionally, you have to specify the sample identifier column in the sample annotation sheet that determines the samples in both the genotyping and DNA methylation data. For larger files, we recommend to activate the option to store large matrices on disk rather than in main memory (`hdf5dump`).

Please note that we provide an accompanying R-package (methQTL.data) comprising additional information including microarray annotations and test data sets for this vignette. Furthermore, we provide a patched version of CRLMM that supports the newer Illumina genotyping microarrays (Illumina OmniExome and Illumina OmniExpress) from GitHub.

```
if(!requireNamespace("methQTL.data")){
  devtools::install_github("MPIIComputationalEpigenetics/methQTL-package.data")
}
library(methQTL.data)
idat.geno <- system.file("extdata",package="methQTL.data")
meth.mat <- system.file("extdata/meth_matrix.tsv.gz",package="methQTL.data")
anno.sheet <-  system.file("extdata/sample_annotation.tsv",package="methQTL.data")
qtlSetOption(hdf5dump=TRUE,
    meth.data.type="data.files",
    geno.data.type="idat")
options(fftempdir="tmp/")
imp.data <- doImport(data.location = c(idat.dir=meth.mat,geno.dir=idat.geno),
                    s.anno = anno.sheet,
                    s.id.col = "geo_accession",
            out.folder = "out_dir",
                    tab.sep = "\t",
            idat.platform="humanomni258v1p1b")
```

For imputed data, no further processing is performed on the genotyping data and the dosage values are used as they are:

```
idat.dir <- "idat_dir"
geno.dir <- "geno_dir"
anno.sheet <- "sample_annotation.tsv"
qtlSetOption(hdf5dump=TRUE)
```

```
imp.data <- doImport(data.location = c(idat.dir=idat.dir,geno.dir=geno.dir),
                     s.anno = anno.sheet,
                     s.id.col = "ID",
                     tab.sep = "\t",
                     out.folder = getwd())
```

Please note that the `recode.allele.frequencies` option specifies, if, according to the cohort analyzed, SNP reference and alternative allele are to be recoded according to the allele frequencies in the available samples. Alternatively, a path to a local version of dbSNP [@Sherry2001] can be provided through `db.snp.ref`, and reference/alternative allele information will be automatically parsed from the database. This is especially crucial, if imputation is to be performed, since the Michigan Imputation Server is sensitive to reference mismatches.

methQTL calling

Although *MAGAR* conceptually splits the methQTL calling into two steps ((i) compute correlation block, (ii) call methQTL per correlation block), only a single function call is needed. The function only requires the input `methQTLInput` object produced in the previous step, but further options, such as covariates and the p-value cutoff can be directly specified as a function parameter, or as global parameters using `?qtlSetOption`.

```
qtlSetOption(standard.deviation.gauss=100,
    cluster.cor.threshold=0.75)
meth.qtl.res <- doMethQTL(imp.data,default.options=F,p.val.cutoff=1e-3)
```

We will now present the two steps of the methQTL calling procedure in more detail.

Compute CpG correlation blocks

Since neighboring CpGs are often highly correlated, using each CpG independently as a potential methQTL candidate leads to many redundant results. We thus aimed to approximate *DNA methylation haplotypes* by determining highly correlated CpGs in close vicinity. The procedure itself is split into six steps, and is performed for each chromosome independently:

1. Compute the (Pearson) correlation matrix between all CpGs (futher correlation types available in option `correlation.type`)
2. Construct the distance matrix from the correlation matrix
3. Discard all interactions with a correlation lower than a given threshold (option: `cluster.cor.threshold`)
4. Weight the distance according to the genomic distance between the two CpGs with a Gaussian (option: `standard.deviation.gauss`). Higher values for the standard deviation lead to a lower penalty on distal CpGs, thus the clusters will become larger.

5. Discard all interactions ranging longer than the option `absolute.distance.cutoff`
6. Compute the Louvain clustering on the undirected, weighted graph induced by the distance matrix

This will return a clustering according to the correlation structure between neighboring CpGs that we will later use for methQTL calling. Note that we used simultation experiments to determine the parameters for each data type individually. They will be automatically loaded for the dataset that is used and are:

- **450k:** `cluster.cor.threshold`=0.2, `standard.deviation.gauss`=5,000, `absolute.distance.cutoff`=500,000
- **EPIC:** `cluster.cor.threshold`=0.2, `standard.deviation.gauss`=3,000, `absolute.distance.cutoff`=500,000
- **RRBS/WGBS:** `cluster.cor.threshold`=0.2, `standard.deviation.gauss`=250, `absolute.distance.cutoff`=500,000

Call methQTL per correlation block

From the list of correlation blocks, *MAGAR* computes methQTL interactions with all SNPs on the same chromosome. The process is split into three steps:

1. Compute a representative CpG (tag-CpG) per correlation block, as specified with the option `representative.cpg.computation` (default: *row.medians*).
2. Discard all SNPs that are further than `absolute.distance.cutoff` (default: 1,000,000) away from the representative CpG
3. Call methQTL by using linear models. Multiple options of methQTL calling are available and can be selected via the option `linear.model.type` (default: *classical.linear*). Alternatively, *fastQTL* can be set as an option for `meth.qtl.type`. This will tell the package to use the fastQTL software [@Ongen2016].

The `meth.qtl.type` tells, how a methQTL interaction is defined and provides three options, in addition to the already mentioned *fastQTL*:

1. *oneVSall*: A CpG can only be influenced by one SNP. We choose the one with the lowest p-value.
2. *twoVSall*: A CpG can both positively and negatively be influenced by two independent SNPs. The package will output those fulfilling the p-value cutoff.
3. *allVSall*: For each CpG, all SNPs showing a p-value lower than the p-value cutoff will be returned.

In the methQTL calling process, potential covariates can be specified using the option *sel.covariates*. We recommend to include at least *age* and *sex* as covariates, as they have a strong influence on the DNA methylation pattern.

Downstream analysis and interpretation

How to use *methQTLResult*

The above procedure will create an object of class `methQTLResult`, which contains the methQTL that are called in the previous step. To get a table of all the methQTL, you need to extract the information from the object. In the majority of the function calls below, there is the option `type`, which takes on the values: * 'SNP': To characterize the SNPs that influence any DNA methylation state * 'CpG': To characterize the representative CpGs per correlation block that are influences by any SNP genotype * 'cor.block': To characterize all CpGs, which are part of a correlation block, whose representative CpG is influenced by any genotype

Furthermore, you can obtain genomic annotations for both the CpGs and the SNPs involved in the methQTL interactions:

```
result.table <- getResult(meth.qtl.res)
head(result.table)
anno.meth <- getAnno(meth.qtl.res,"meth")
head(anno.meth)
anno.geno <- getAnno(meth.qtl.res,"geno")
head(anno.geno)
```

For more detailed information about the output, also see the function `getResultsGWASMap`.

Plots

To visualize methQTL, *MAGAR* provides some plotting functions. Most functions return an object of type `ggplot`, which can be subsequently stored or viewed. Either all methQTL can be simultaneously visualized in a single plot, or a specific methQTL can be visualized:

```
result.table <- result.table[order(result.table$P.value,decreasing=F),]
qtlPlotSNPCpGInteraction(imp.data,result.table$CpG[1],result.table$SNP[1])
qtlDistanceScatterplot(meth.qtl.res)
```

Interpretation functions

The package provides a bunch of interpretation functions to characterize the detected methQTLs. This includes LOLA enrichment analysis[@LOLA] (`qtlLOLAEnrichment`), genomic annotation enrichment based on putative regulatory elements defined by the Ensembl Regulatory Build[@Zerbino2015] (`qtlAnnotationEnrichment`), enrichment analysis of different base substitutions in SNPs (`qtlBaseSubstitutionEnrichment`), or TFBS motif enrichment using TFBSTools. Enrichment is compared for the methQTLs that are available in the provided `methQTLResult` (for a single input), or to the overlapping QTLs for a list of `methQTLResult`. The background of the enrichment is defined as all the SNPs/CpGs that have been used as input to the methQTL calling.

```r
res <- qtlBaseSubstitutionEnrichment(meth.qtl.res)
qtlPlotBaseSubstitution(meth.qtl.res,merge=TRUE)
```

Lists of methQTL results

Most of the functions discussed above either support a single `methQTLResult` as input, or a list of such objects. In case a list is specified, the functions will typically overlap the methQTLs found and compare those with all SNPs/CpGs that have been used for methQTL calling. Additionally, there are functions that particularly work on a list of `methQTLResult` objects and that perform overlapping, or determine the methQTLs specific to a dataset.

```r
meth.qtl.list <- list(First=meth.qtl.res.1,Second=meth.qtl.res.2,Third=meth.qtl.res.3)
qtlVennPlot(meth.qtl.list)
qtlUpsetPlot(meth.qtl.list,type = "cor.block")
spec.first <- getSpecificQTL(meth.qtl.list$First,meth.qtl.list[-1])
```

Advanced configuration

Employ *MAGAR* on a scientific compute cluster

MAGAR can automatically distribute jobs across a high performance compute cluster, which has been setup using the Sun Grid Engine (SGE) technology. You can pass the option `cluster.submit` to `doMethQTL` and thus activate the cluster submission. Note that you'll also have to specify a path to an executable *Rscript* and potentially specify resource requirements using the option setting `cluster.config`.

```r
qtlSetOption(cluster.config = c(h_vmem="60G",mem_free="20G"))
qtlSetOption(rscript.path = "/usr/bin/Rscript")
meth.qtl.res <- doMethQTL(meth.qtl = imp.data,
                          cluster.submit = T)
```

References