Users' Guide
MORE

Sonia Tarazona
Blanca Tomás Riquelme

September 11, 2018

Contents

1 Introduction **2**

2 Getting started **3**

3 Input data **3**

4 Generating the regression models with MORE **5**
4.1 GetGLM input parameters 5
4.2 GetGLM output 7
4.3 Running an example 8

5 Retrieving significant regulations from MORE results **10**
5.1 RegulationPerCondition input parameters 10
5.2 Interpreting RegulationPerCondition output with an example 11

6 Plotting MORE results **12**
6.1 plotGLM input parameters 12
6.2 Interpretation of MORE plots 13

7 How to use MORE with R Shiny **17**
7.1 MORE Shiny application example 17

8 How to cite MORE package **20**

1 Introduction

One of the most common questions to be addressed when performing a multi-omics experiment is how the levels of given biological entities are being regulated by other biological entities under certain conditions. An example of this type of study would be understanding the regulatory mechanisms behind the changes in gene expression.

Potential regulators of a given gene such as miRNAs, transcription factors (TF), methylation sites, etc. can be either retrieved or predicted from public databases or obtained by a combination of experimental and computational procedures. However, a methodology for selecting the specific regulators of a particular biological system studied under certain experimental conditions is required. This is the goal of the MORE (Multi-Omics REgulation) method: modelling gene expression as a function of experimental variables, such as diseases or treatments, and the potential regulators of a given gene. The idea is to obtain more specific candidate regulators for the biological system under study by applying regression models, specifically, generalized linear models (GLM). MORE facilitates the application of GLM to multi-omic data and, although it was originally conceived to study gene expression regulation, its usage can be extended to protein or metabolite levels, for instance.

MORE requires several data inputs: gene expression data, regulators omic data, experimental design and potential associations between genes and regulators. With this input data, MORE generates the initial model equation, which is different for each gene because each one of them has different potential regulators. MORE admits numerical omic data (continuous or discrete data) or binary data.

It is strongly recommended to fit MORE models only to genes that present significant changes in any of the experimental conditions studied, that is, to differentially expressed genes (DEGs). DEGs can be selected with the standard procedures depending on the experimental design but DEGs selection is not included in the MORE algorithm and must be done by the user.

This idea can be extended to potential regulators, since regulators that do not change accross conditions are not good candidates to regulate gene expression. Removing non-DE regulators will also help to reduce the number of predictors in the model since an excess of them would prevent the estimation of regression coefficients. Even so, MORE has several funcionalities to filter regulators with missing values or with low variation, highly correlated regulators, and to perform variable selection.

MORE package also includes a function to retrieve the significant regulations and the magnitude of the regulatory effect under each experimental condition considered, and an additional function to graphically investigate the relationship between genes and regulators.

2 Getting started

The MORE method is available as an R package from https://bitbucket.org/ConesaLab/more. As for other packages in bitbucket, it can be installed from R with the following instructions:

```
> install.packages("devtools")
> devtools::install_bitbucket("ConesaLab/more")
```

3 Input data

This section describes the main data files required by MORE to generate the regression models.

Gene expression data Expression values for each gene, in rows, under each experimental condition or replicate, in columns. MORE accepts either a **matrix** or a **data frame**. See an example below:

```
> head(TestData$GeneExpressionDE)
                    Group1.Time1.Rep1 Group1.Time2.Rep1 Group1.Time3.Rep1 Group1.Time4.Rep1
ENSMUSG00000000078              7366              7408              6081              7045
ENSMUSG00000056999              1494              2742              3749              4070
ENSMUSG00000024873               294               323               660               757
ENSMUSG00000015461              6122              6289              6914              7633
ENSMUSG00000058135               733              3927              7317              8337
ENSMUSG00000038208              1042              1069              1052              1021
                    Group1.Time5.Rep1 Group1.Time6.Rep1 Group1.Time7.Rep1 Group1.Time8.Rep1
ENSMUSG00000000078              6717              7893              7696              8072
ENSMUSG00000056999              4750              5595              7721              6896
ENSMUSG00000024873               714               665               866               950
ENSMUSG00000015461              6944              5459              4752              2783
ENSMUSG00000058135              8853              7306              5892              3549
ENSMUSG00000038208              1088              1055              1024              1098
                    Group2.Time1.Rep1 Group2.Time2.Rep1 Group2.Time3.Rep1 Group2.Time4.Rep1
ENSMUSG00000000078              5920              4956              5434              4357
ENSMUSG00000056999              1618              4081              5749              7849
ENSMUSG00000024873               243               486               440               425
ENSMUSG00000015461              2304               724               108                 1
ENSMUSG00000058135              1947              4318              7141              7096
ENSMUSG00000038208              1138              1205              1039              1372
                    Group2.Time5.Rep1 Group2.Time6.Rep1 Group2.Time7.Rep1 Group2.Time8.Rep1
ENSMUSG00000000078              5405              4964              6762              5500
ENSMUSG00000056999              7133              7744              6763              6095
ENSMUSG00000024873               522               592               899               848
ENSMUSG00000015461               276              1315              2755              4159
ENSMUSG00000058135              7717              6573              7212              3675
ENSMUSG00000038208              1444              1111              1046               885
```

Experimental design Matrix or data frame containing the experimental covariates, such as treatments, diseases, strains, dose of a drug, etc. The object must have as many rows as the number of columns in gene expression data, as below. There is no

restriction for the number of columns but it must be taken into account that MORE will combine all the experimental covariates into a single variable. For instance, in the example below, the new single covariate would combine Time and Group2 to obtain the values: 1_0, 2_0, ...,7_1, 8_1. Therefore, in this case, it makes more sense excluding Time from the experimental design and just including the covariate Group2.

```
> TestData$edesign
                   Time Group2
Group1.Time1.Rep1     1      0
Group1.Time2.Rep1     2      0
Group1.Time3.Rep1     3      0
Group1.Time4.Rep1     4      0
Group1.Time5.Rep1     5      0
Group1.Time6.Rep1     6      0
Group1.Time7.Rep1     7      0
Group1.Time8.Rep1     8      0
Group2.Time1.Rep1     1      1
Group2.Time2.Rep1     2      1
Group2.Time3.Rep1     3      1
Group2.Time4.Rep1     4      1
Group2.Time5.Rep1     5      1
Group2.Time6.Rep1     6      1
Group2.Time7.Rep1     7      1
Group2.Time8.Rep1     8      1
```

Regulatory omic data This object must be a list where each element is a matrix or data frame containing the data for each "regulatory" omic (miRNA expression, transcription factor expression, etc.), with structure similar to gene expression data: regulators in rows and experimental conditions in columns (the columns must be the same as in gene expression and in the same order). See example below (TestData$data.omics$`miRNA-seq`).

```
> head(TestData$data.omics$`miRNA-seq`)
              Group1.Time1.Rep1 Group1.Time2.Rep1 Group1.Time3.Rep1 Group1.Time4.Rep1 Group1.Time5.Rep1 Group1.Time6.Rep1
mmu-miR-125a-5p              7100              7229             10066              8868             12649             16321
mmu-miR-141-5p                 19                49                40               107               176               183
mmu-miR-145a-3p            117329             85793             73670             57547             43926             21426
mmu-miR-148b-3p            125403            136517            137890            190417            150718            217291
mmu-miR-150-5p               1256              1098               707              1025               599               253
mmu-miR-152-3p                344               445               322               529               655              1082
              Group1.Time7.Rep1 Group1.Time8.Rep1 Group2.Time1.Rep1 Group2.Time2.Rep1 Group2.Time3.Rep1 Group2.Time4.Rep1
mmu-miR-125a-5p             14760             15164              1655               166                 8               562
mmu-miR-141-5p                168               343                58                 8                 2                 9
mmu-miR-145a-3p             11769                 2             11593              1539               125              7075
mmu-miR-148b-3p            203966            203880            108489            116911            102194             94160
mmu-miR-150-5p                190                 0              1289               844               617               517
mmu-miR-152-3p               1645              1441                63                17                 0               121
              Group2.Time5.Rep1 Group2.Time6.Rep1 Group2.Time7.Rep1 Group2.Time8.Rep1
mmu-miR-125a-5p              2818              4997              7528              9159
mmu-miR-141-5p                 13               117               121               283
mmu-miR-145a-3p             19253             32778             61536            106987
mmu-miR-148b-3p            126998            150972            171944            205767
mmu-miR-150-5p                374               370               164                 0
mmu-miR-152-3p                 94               270               665               687
```

Associations For each regulatory omic, associations between regulators and genes which indicate which are the potential regulators of each gene that will be consequently incorporated into the initial equation of the regression model. The association objects must be data frames and stored in a single **list**. The names of the

elements of this list must be the names of the list collecting regulatory omic data (see TestData$associations$‘miRNA-seq‘ for the corresponding example).

```
> head(TestData$associations$`miRNA-seq`)
                  ID            Gene
1 ENSMUSG00000024873  mmu-miR-335-3p
2 ENSMUSG00000024873 mmu-miR-1912-3p
3 ENSMUSG00000024873  mmu-miR-615-5p
4 ENSMUSG00000024873  mmu-miR-322-5p
5 ENSMUSG00000024873 mmu-miR-1894-3p
6 ENSMUSG00000024873 mmu-miR-7082-3p
```

4 Generating the regression models with MORE

The **GetGLM** function in MORE adjusts a generalized linear model (GLM) for each gene (protein, metabolite etc.) in the *GeneExpresion* object, to determine which regulators and experimental variables have a significant effect on the response variable (gene expression, protein levels, etc.). These are the arguments accepted by the function, which are described in detail in Section 4.1.

```
GetGLM(GeneExpression, associations, data.omics, edesign = NULL,
center = TRUE, scale = FALSE, Res.df = 5, epsilon = 0.00001,
alfa = 0.05, MT.adjust = "none", family = negative.binomial(theta=10),
elasticnet = 0.5, stepwise = "backward", interactions.reg = TRUE,
min.variation = 0, correlation = 0.9, min.obs = 10, omic.type = 0)
```

4.1 GetGLM input parameters

GeneExpression Matrix or data frame containing gene expression data with genes in rows and experimental samples in columns. The row names must be the gene IDs.

associations List where each element corresponds to a different omic data type (miRNAs, transcription factors, methylation, etc.). The names of the list will be the omics. Each element is a data frame with two columns (optionally three) describing the potential interactions between genes and regulators for that omic. First column must contain the regulators, second the gene IDs, and an additional column can be added to describe the type of interaction (for example, in methylation data, if a CpG site is located in the promoter region of the gene, in the first exon, etc.).

data.omics List where each element corresponds to a different omic data type (miRNAs, transcription factors, methylation, etc.). The names of this list will be the omics and each element of the list is a matrix or data frame with omic regulators in rows and samples in columns.

edesign Data frame or matrix describing the experimental design. Rows must be the samples, that is, the columns in the **GeneExpression**, and columns must be the experimental variables to be included in the model, such as disease, treatment, etc.

center If TRUE (default), the omic data are centered.

scale If TRUE, the omic data are scaled. Default value is set to FALSE.

Res.df Number of degrees of freedom in the residuals. By default, 5. Increasing **Res.df** will increase the power of the statistical model and decrease the number of significant predictors.

epsilon A threshold for the positive convergence tolerance in the GLM model. By default, 0.00001.

alfa Significance level. By default, 0.05.

MT.adjust Multiple testing correction method to be used within the stepwise variable selection procedure. By default, "none". You can see the different options in `?p.adjust`.

family Error distribution and link function to be used in the model (see `glm` for more information). By default, `negative.binomial(theta = 10)`.

elasticnet ElasticNet mixing parameter. By default, 0.5. These are the values that can be passed to this argument:

NULL No ElasticNet variable selection is performed.

Value between 0 and 1 ElasticNet is applied with this number being the combination between ridge and lasso penalization.

Value 0 The ridge penalty.

Value 1 The lasso penalty.

stepwise Stepwise variable selection method to be applied. It can be one of: "none", "backward" (by default), "forward", "two.ways.backward" or "two.ways.forward".

interactions.reg If TRUE (default), MORE allows for interactions between each regulator and the experimental covariate.

min.variation For numerical regulators, the minimum change that a regulator must present across conditions to keep it in the regression models. For binary regulators, if the proportion of the most repeated value equals or exceeds this value, the regulator will be considered to have low variation and removed from the regression models. By default, its value is 0.

correlation Correlation threshold (in absolute value) to decide which regulators are correlated, in which case, a representative of the group of correlated regulators is chosen to enter the model. By default, 0.9.

min.obs Minimun number of observations a gene must have to compute the GLM model. By default, 10.

omic.type Vector with as many elements as the number on omics, which indicates if the omic values are numeric (0, default) or binary (1). When a single value is given, the type for all the omics is set to that value. By default, 0.

4.2 GetGLM output

The object returned by the GetGLM function is a list that contains the following elements:

ResultsPerGene is a list with as many elements as genes in GeneExpression object. For each gene, there is a list containing the following information:

Y Data frame with the response variable values for that gene (y), the values fitted by the model (fitted.y), and the residuals of the GLM (residuals).

X Data frame with all the predictors included in the final model.

coefficients Matrix with the estimated coefficients for significant regulators and their p-values.

allRegulators Data frame with all the initial potential regulators in rows and the following information in columns: gene, regulator, omic, area (the third optional column in associations), filter (if the regulator has been filtered out of the model, this column indicates the reason), and Sig (1 if the regulator is significant and 0 if not). Regarding the filter column, several values are possible:

- *MissingValue*: If the regulator has been filtered out of the study because it has missing values.
- *LowVariation*: If the regulator has been filtered out of the study because it has lower variability than the threshold set by the user in min.variation parameter.
- *ElasticNet*: If the regulator has been excluded of the model by the Elastic-Net variable selection procedure.
- *Model*: When the regulator is included in the initial equation model.
- *omic_mcX_X*: For instance, *TF_mc1_R*. This notation is related to highly correlated regulators and how they are treated to avoid the multicollinearity problem. Following the *TF_mc1_R* example, it means that there are two or more transcription factors (TF omic) potentially regulating the gene that are highly correlated (in absolute value). In such cases, one of them is chosen as the representative and indicated with _R. The rest of them are labeled with _P if positively correlated with the representative and with _N

if negatively correlated. An additional row is then added to this table, with the regulator *TF_mc1_R* and the filter label being *Model*, since only this representative is considered in the model. When there are several groups of correlated regulators for the same omic, it is indicated with _mc1_, _mc2_, etc.

significantRegulators A character vector containing the significant regulators.

GlobalSummary List that contains the following elements:

GoodnessOfFit Matrix that collects the model p-value, the final number of degrees of freedom in the residuals, the R-squared value (which for GLMs is defined as the percentage of deviance explained by the model) and the Akaike information criterion (AIC) value for all the genes studied.

ReguPerGene Matrix containing, for each omic and gene, the number of initial regulators, the number of regulators included in the initial model and the number of significant regulators.

GenesNOmodel List of genes for which the final GLM model could not be obtained. There are three possible reasons for that and they are indicated: "Too many missing values", "No predictors after EN" and "GLM error", where EN refers to the ElasticNet variable selection procedure.

Arguments List with the arguments used to generate the model: experimental design matrix, minimum degrees of freedom in the residuals, significance level, family distribution, variable selection, etc.

4.3 Running an example

An example of the execution of **GetGLM** function is shown next by using simulated data. The data file `TestData.RData` is available in the package, so the user can test and visualize these results.

```
> data("TestData")
```

In this file, the gene expression matrix corresponds to the omic RNA-seq (**GeneExpressionDE**) and there is a list with three matrices of regulators in the **data.omics** object:

ChIP-seq Omic that contains binary values (1 if the TF analysed in the ChIP-seq experiment binds to the gene and 0 if not).

miRNA-seq miRNA expression data (count data).

TF TF expression data (count data).

Therefore, the **OmicType** parameter is defined as

```
> OmicType = c(1, 0, 0)
> names(OmicType) = names(data.omics)
```

The experimental design matrix (**edesign**) consists of 8 time points in two conditions, which results in a total of 16 experimental samples, but time is not to be considered as an experimental covariate since we are interested in comparing temporal profiles for the two experimental groups.

We can run the following **GetGLM** code to obtain the regression models for our genes:

```
> SimGLM = GetGLM(GeneExpression = GeneExpressionDE,
associations = associations, data.omics = data.omics,
edesign = edesign[,-1, drop = FALSE], Res.df = 7,
epsilon = 0.00001, alfa = 0.05, MT.adjust = "fdr",
family = negative.binomial(theta = 10), elasticnet = 0.3,
stepwise = "backward", interactions.reg = TRUE,
correlation = 0.9, min.variation = NULL,
min.obs = 10, omic.type = OmicType)
```

The estimated coefficients of the significant regulators in the final model GLM for the gene ENSMUSG00000000078 and their associated p-values are

```
> SimGLM$ResultsPerGene$ENSMUSG00000000078$coefficients
                          coefficient      p-value
(Intercept)              8.768718e+00 2.043440e-23
`mmu-miR-18a-5p`         2.549854e-06 4.122175e-04
Hmga1                   -1.099662e-05 5.564550e-03
Mef2d                   -2.381339e-05 2.018984e-04
TF_mc1_R                -1.339656e-04 4.201164e-03
`Group1:mmu-miR-431-5p`  3.675612e-06 1.273422e-03
```

The **allRegulators** table shows, for each gene, their regulators, omic, area, the kind of filter applied and the significance of the regulator. In this case (see **filter** column), in ChIP-seq, the regulators are correlated, so a random representative regulator (R) has been chosen. In addition, it is indicated if the correlation with the representative is negative (N) or positive (P). Other possible filters are also indicated as `ElasticNet`, `LowVariation`... while `Model` means that the regulator was included in the model. On the other hand, the **Sig** column returns 1 if the regulator was significant in the final model and 0 if not.

```
> SimGLM$ResultsPerGene$ENSMUSG00000000078$allRegulators[1:10,]
                                gene           regulator      omic area          filter Sig
13_5789384_5789613 ENSMUSG00000000078 13_5789384_5789613  ChIP-seq       ChIP-seq_mc1_P   0
13_5803679_5803880 ENSMUSG00000000078 13_5803679_5803880  ChIP-seq       ChIP-seq_mc1_P   0
13_5804696_5804875 ENSMUSG00000000078 13_5804696_5804875  ChIP-seq       ChIP-seq_mc1_P   0
13_5860779_5861047 ENSMUSG00000000078 13_5860779_5861047  ChIP-seq       ChIP-seq_mc1_N   0
13_5861542_5861793 ENSMUSG00000000078 13_5861542_5861793  ChIP-seq       ChIP-seq_mc1_R   0
13_5926774_5926959 ENSMUSG00000000078 13_5926774_5926959  ChIP-seq       ChIP-seq_mc1_P   0
mmu-miR-431-5p     ENSMUSG00000000078     mmu-miR-431-5p miRNA-seq                Model   1
mmu-miR-18a-5p     ENSMUSG00000000078     mmu-miR-18a-5p miRNA-seq                Model   1
mmu-miR-18b-5p     ENSMUSG00000000078     mmu-miR-18b-5p miRNA-seq           ElasticNet   0
mmu-miR-25-3p      ENSMUSG00000000078      mmu-miR-25-3p miRNA-seq           ElasticNet   0
>
> SimGLM$ResultsPerGene$ENSMUSG00000000078$allRegulators[90:105,]
                           gene      regulator     omic area       filter Sig
Satb1           ENSMUSG00000000078          Satb1       TF        ElasticNet   0
Tbp             ENSMUSG00000000078            Tbp       TF          TF_mc1_N   1
Zfp628          ENSMUSG00000000078         Zfp628       TF          TF_mc1_N   1
Zfp64           ENSMUSG00000000078          Zfp64       TF        ElasticNet   0
Cux1            ENSMUSG00000000078          Cux1       TF      LowVariation   0
Gtf2b           ENSMUSG00000000078          Gtf2b       TF      LowVariation   0
Mafk            ENSMUSG00000000078          Mafk       TF      LowVariation   0
Pou6f1          ENSMUSG00000000078         Pou6f1       TF      LowVariation   0
Rara            ENSMUSG00000000078          Rara       TF      LowVariation   0
Rfx1            ENSMUSG00000000078          Rfx1       TF      LowVariation   0
Rreb1           ENSMUSG00000000078         Rreb1       TF      LowVariation   0
Rxra            ENSMUSG00000000078          Rxra       TF      LowVariation   0
Zfp692          ENSMUSG00000000078         Zfp692       TF      LowVariation   0
Zfp775          ENSMUSG00000000078         Zfp775       TF      LowVariation   0
ChIP-seq_mc1_R ENSMUSG00000000078 ChIP-seq_mc1_R ChIP-seq        ElasticNet   0
TF_mc1_R        ENSMUSG00000000078       TF_mc1_R       TF             Model   1
```

5 Retrieving significant regulations from MORE results

The function **RegulationPerCondition** is applied to the **GetGLM** output and returns a summary table that contains all the significant regulations, that is, all the pairs gene-regulator that were significant in MORE models. Moreover, it provides the regression coefficient that relates the gene and the regulator for each experimental condition after testing if this coefficient is significant or not.

```
RegulationPerCondition(getGLMoutput, betaTest = TRUE)
```

5.1 RegulationPerCondition input parameters

getGLMoutput Object containing the output of **getGLM** function.

betaTest If TRUE (default), a hypothesis test is perfomed on the coefficients of the final GLM model. If the coefficients are significant, their values are displayed. If not, it is indicated with a 0.

5.2 Interpreting RegulationPerCondition output with an example

Following with the previous example, we can run the **RegulationPerCondition** function with the argument **betaTest** equal to TRUE:

```
> myresults = RegulationPerCondition(SimGLM)
```

The output is the following table where only the last nine pairs gene-regulator are shown:

```
> myResults = RegulationPerCondition(SimGLM)
> myResults[1:20,]
```

	gene	regulator	omic	area	representative	Group0	Group1
mmu-miR-18a-5p	ENSMUSG00000000078	mmu-miR-18a-5p	miRNA-seq			2.550e-06	2.550e-06
Hmga1	ENSMUSG00000000078	Hmga1	TF			-1.100e-05	-1.100e-05
Mef2d	ENSMUSG00000000078	Mef2d	TF			-2.381e-05	-2.381e-05
mmu-miR-431-5p	ENSMUSG00000000078	mmu-miR-431-5p	miRNA-seq			0.000e+00	3.676e-06
Arid5a	ENSMUSG00000000078	Arid5a	TF		Rfxank	-1.340e-04	-1.340e-04
Rfxank	ENSMUSG00000000078	Rfxank	TF		Rfxank	-1.340e-04	-1.340e-04
Tbp	ENSMUSG00000000078	Tbp	TF		Rfxank	1.340e-04	1.340e-04
Zfp628	ENSMUSG00000000078	Zfp628	TF		Rfxank	1.340e-04	1.340e-04
mmu-miR-668-5p	ENSMUSG00000056999	mmu-miR-668-5p	miRNA-seq			1.453e-05	1.453e-05
mmu-miR-6931-5p	ENSMUSG00000056999	mmu-miR-6931-5p	miRNA-seq			0.000e+00	-3.536e-05
mmu-miR-543-5p	ENSMUSG00000056999	mmu-miR-543-5p	miRNA-seq			0.000e+00	-2.643e-05
mmu-miR-7648-3p	ENSMUSG00000024873	mmu-miR-7648-3p	miRNA-seq			-1.276e-05	-1.276e-05
mmu-miR-6931-5p1	ENSMUSG00000024873	mmu-miR-6931-5p	miRNA-seq		mmu-miR-6931-5p	-2.059e-05	0.000e+00
mmu-miR-145a-3p	ENSMUSG00000024873	mmu-miR-145a-3p	miRNA-seq		mmu-miR-6931-5p	-2.059e-05	0.000e+00
mmu-miR-376c-3p	ENSMUSG00000015461	mmu-miR-376c-3p	miRNA-seq			3.343e-05	3.343e-05
Arnt	ENSMUSG00000015461	Arnt	TF			3.965e-04	3.965e-04
Arnt1	ENSMUSG00000058135	Arnt	TF			-7.829e-05	-7.829e-05
Sp4	ENSMUSG00000058135	Sp4	TF			-1.568e-04	-1.568e-04
Ep300	ENSMUSG00000058135	Ep300	TF		Zfp513	-3.759e-04	-3.759e-04
Zfp513	ENSMUSG00000058135	Zfp513	TF		Zfp513	-3.759e-04	-3.759e-04

This table shows the significant regulators for each gene. The **representative** column indicates if the regulator was chosen as the random representative of a correlated group of regulators or, otherwise, which regulator was taken as the representative of the group. When no information is provided in this column, it means that the regulator was not part of a correlated group of regulators. Regulators correlated positively with the representative will have the same coefficients (same sign) than the representative, while negatively correlated regulators will have the same coefficients of the representative but with opposite sign.

The final columns correspond to the regression coefficients of each regulator for each experimental group. In this case, the experimental design matrix (**edesign**) contained two conditions, so the column `Group0` corresponds to the first condition and `Group1` corresponds to the second one. These are the conclusions we can draw from the coefficients:

- If two experimental groups have the same coefficients, it means that the regulator has the same effect on the gene in both groups.
- If one of the coefficients is 0, it means that the regulator has no effect on the gene under this experimental condition.

- Experimental groups with different non-zero coefficients indicate that the regulator affects the gene in all these experimental groups but the magnitude of the effect is not the same for all these groups.

6 Plotting MORE results

MORE package includes the function **plotGLM** to graphically represent the relationship between genes and regulators: for a given pair gene-regulator, to explore the regulators of a given gene, or to analyze which genes are regulated by a specific regulator.

```
plotGLM(GLMoutput, gene, regulator = NULL, reguValues = NULL,
plotPerOmic = FALSE, gene.col = 1, regu.col = NULL, order = TRUE,
xlab = "", cont.var = NULL, cond2plot = NULL)
```

6.1 plotGLM input parameters

GLMoutput Object generated by the function **getGLM**.

gene ID of the gene to be plotted.

regulator ID of the regulator to be plotted. If NULL (default value), all the regulators of the gene are plotted.

reguValues Vector containing the values of a regulator that the user can optionally provide. If NULL (default value), these values are taken from **GLMoutput** as long as they are available.

plotPerOmic If TRUE, all the significant regulators of the given gene and the same omic are plotted in the same graph. If FALSE (default value), each regulator is plotted in a separate plot.

gene.col Color to plot the gene. By default, 1 (black).

regu.col Color to plot the regulator. If NULL (default), a color will be assigned by the function, that will be different for each regulatory omic.

order If TRUE (default), the values in X axis are ordered.

xlab Label for X axis.

cont.var Vector with length equal to the number of observations in data, which optionally may contain the values of the numerical variable (e.g. time) to be plotted in X axis. By default, NULL.

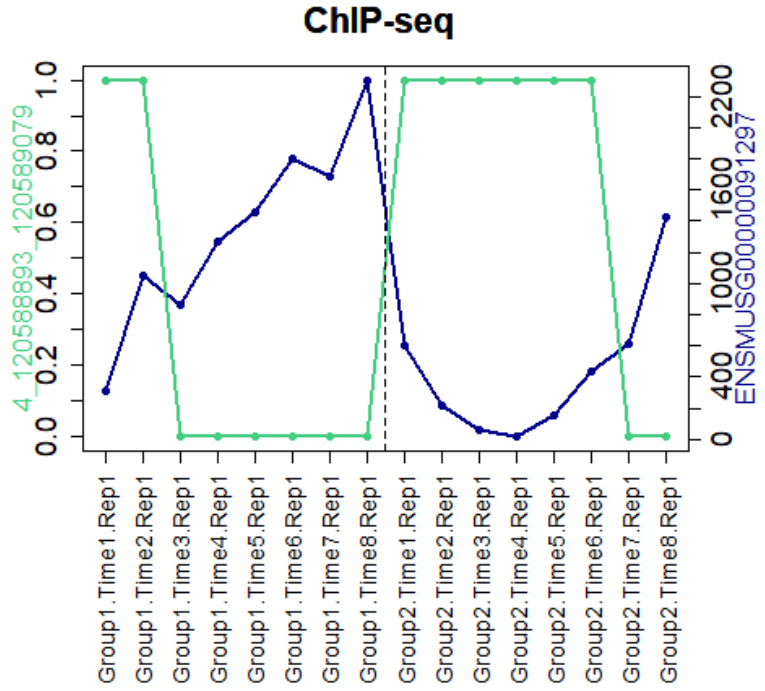
cond2plot Vector or factor indicating the experimental group of each value to represent. If NULL (default), the labels are taken from the experimental design matrix.

6.2 Interpretation of MORE plots

Following the previous example, the MORE graphic below represents the expression profile of a given gene (ENSMUSG00000091297) and the values for a significant regulator of this gene (ChIP-seq regulator 4_120588893_120589079), and can be generated with the following code:

```
> plotGLM(GLMoutput = SimGLM,
          gene = "ENSMUSG00000091297",
          regulator = "4_120588893_120589079",
          plotPerOmic = FALSE,
          gene.col = "blue4",
          regu.col = "seagreen3")
```

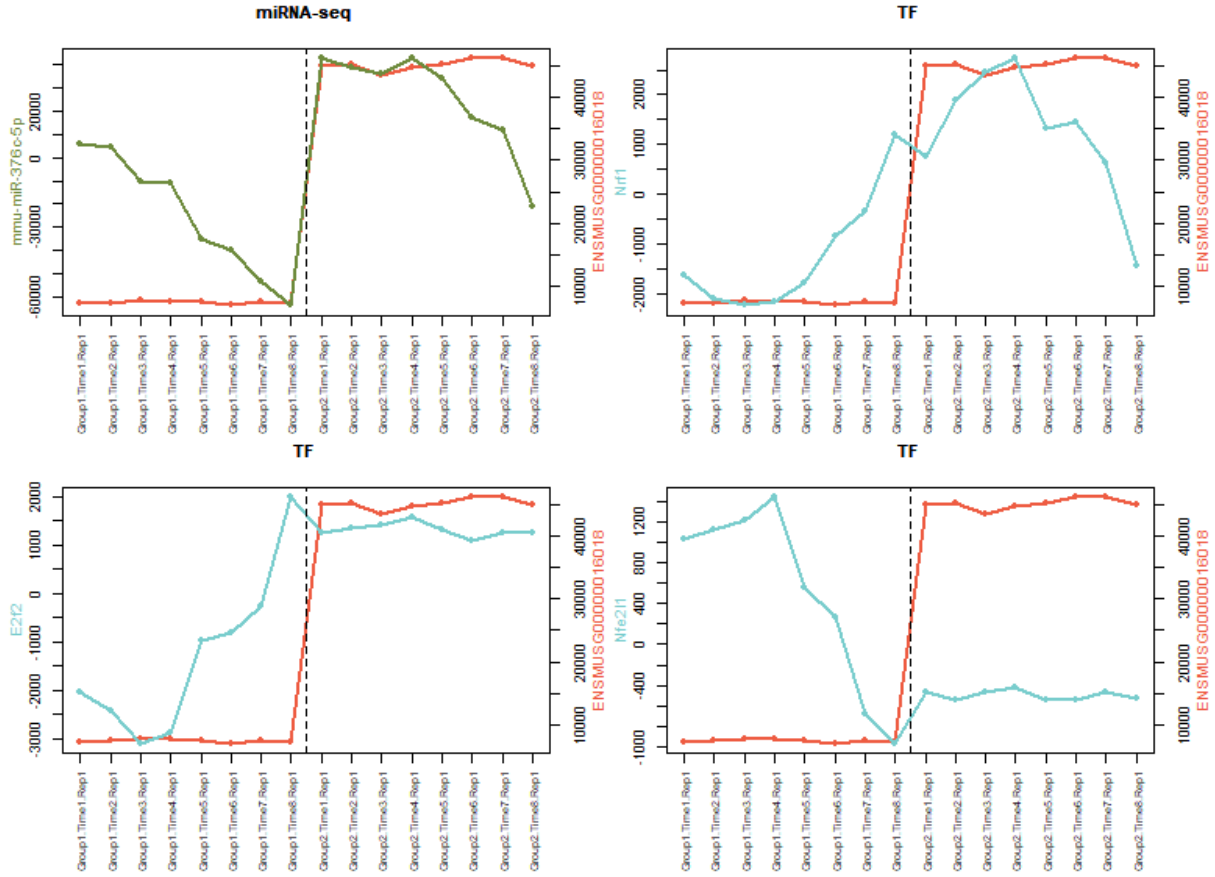
The X axis is divided in the two conditions (1 or 2) and within each condition, the observations are displayed, which correspond to different time points in this case. The right Y axis shows the expression values for the gene (plotted in blue), while the left Y axis indicates the values for the regulator (plotted in green).



If we set the regulator argument to NULL, all the significant regulators of gene ENSMUSG00000016018 will be plotted (4 regulators from two different omics: miRNA-seq and TF).

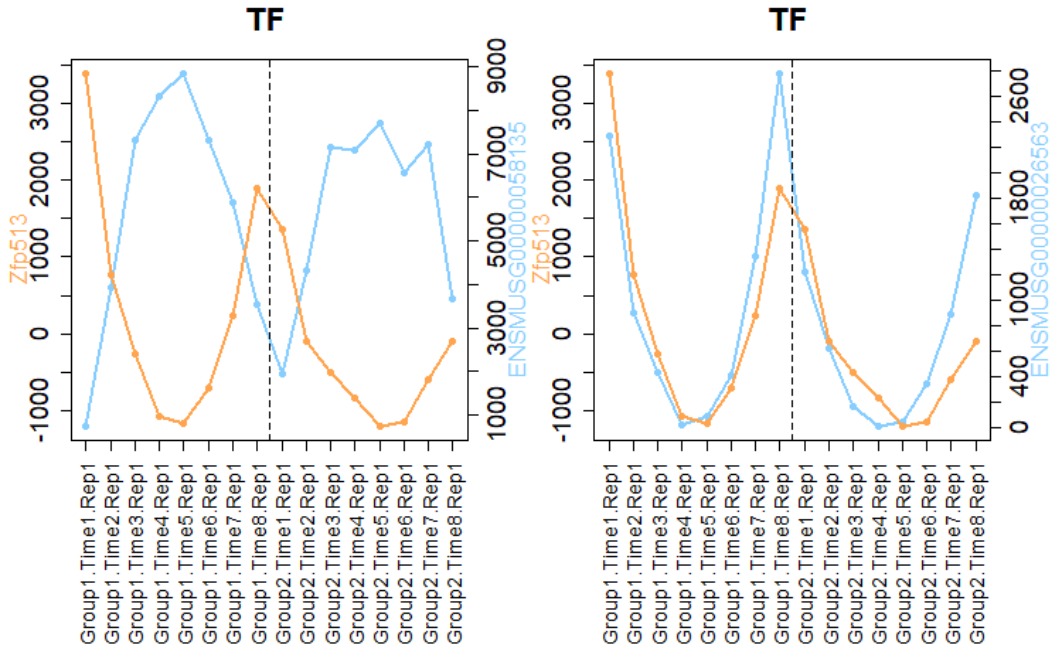
```
> par(mfrow = c(2,2))
> plotGLM(GLMoutput = SimGLM,
          gene = "ENSMUSG00000016018",
          regulator = NULL,
          plotPerOmic = FALSE,
          gene.col = "tomato2")
```

The title of each plot indicates the omic represented in that plot. The values for significant regulators are plotted in different colors according to the omic. The values for the gene are plotted in red, as indicated in the previous code.



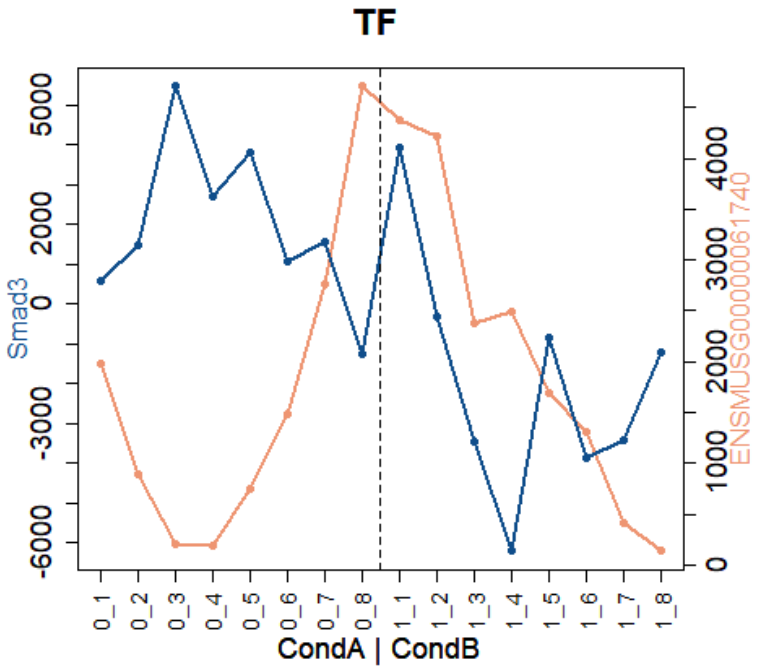
If we want to plot all the genes that are being significantly regulated by a given regulator (e.g. Zfp513), we must set the gene argument to NULL as follows. In this case, the TF regulates two genes: ENSMUSG00000058135 and ENSMUSG00000026563.

```
> par(mfrow = c(1,2))
> plotGLM(GLMoutput = SimGLM,
          gene = NULL,
          regulator = "Zfp513",
          plotPerOmic = FALSE,
          gene.col = "skyblue1",
          regu.col = "tan1")
```



Users can also define their own values for time points with the vector **cont.var**. In addition, they can assign a label for axis X to differentiate between two conditions, A or B, which is the **xlab** parameter. The code and the resulting graph, where the gene is plotted in light orange and the regulator is plotted in blue, can be found below.

```
> plotGLM(GLMoutput = SimGLM,
          gene = "ENSMUSG00000061740",
          regulator = "Smad3",
          plotPerOmic = FALSE,
          gene.col = "lightsalmon2",
          regu.col = "dodgerblue4",
          cont.var = c(1,2,3,4,5,6,7,8),
          xlab = "CondA | CondB")
```



7 How to use MORE with R Shiny

Shiny is an R package that allows for building web applications from R packages or scripts so users that are not familiar with R language can still easily use R packages. We have generated a MORE web application with R Shiny with this purpose. To use the MORE shiny tool, users must first install the Shiny and the Shiny themes packages from CRAN repository with `install.packages()` function.

```
> install.packages("shiny")
> install.packages("shinythemes")
```

Moreover, users must also have previously installed the MORE package described in section 2.

The MORE shiny scripts needed to run the tool are available in the Downloads bitbucket folder for MORE package (https://bitbucket.org/ConesaLab/more/downloads/). There is a file called `app.R`, an example dataset stored in the file `TestDataShiny.RData` and a folder called `www`.

app.R Script to run the web application for MORE method. Users must open this file from RStudio to start using the application .

TestDataShiny.RData Example data file to test the application. We used it as an example of an execution of MORE Shiny.

www Folder containing the style options. Please, do not delete anything in this folder.

Therefore, in order to run the application, please open the `app.R` file, where the MORE method application is located, and execute it using the button `Run App` (the red box in the following picture).

A window will open with the MORE application, as shown in the following figure.

The different options have been explained in the previous sections but, next, we run an example to clarify how to use them. This example can also be visualized in this video: https://youtu.be/SSIaeFRNsXg.

7.1 MORE Shiny application example

Please take into account that MORE Shiny only supports `.RData` files at this time. Once the RData is loaded, the user must indicate the names of the data files in the RData that corresponds to (see the input data defined in the **section 3**): gene expression matrix, experimental design matrix, regulators matrix and association matrix. In summary (for more details see **section 3**):

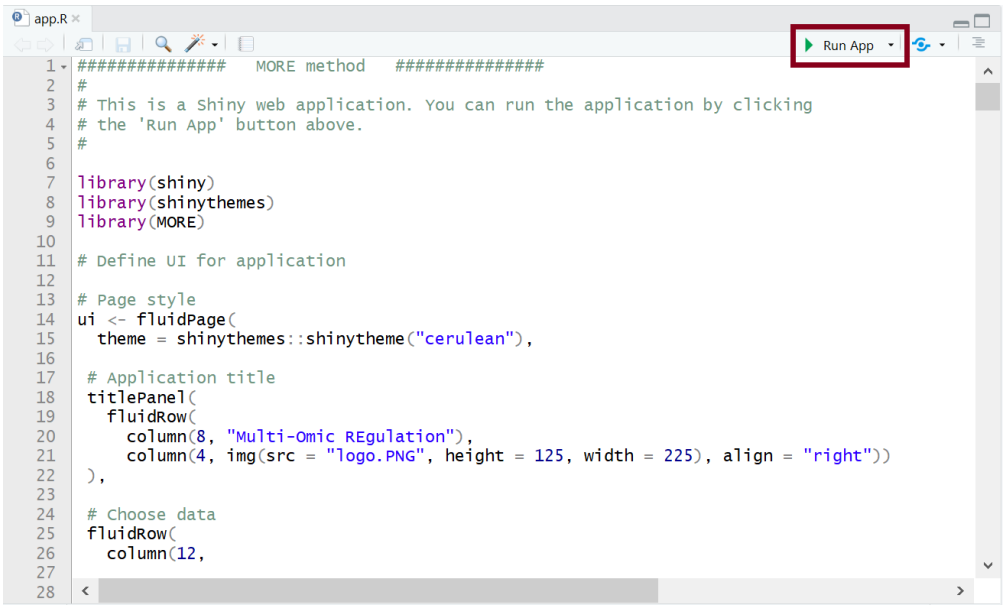


Figure 1: Run MORE application, click `Run App` button

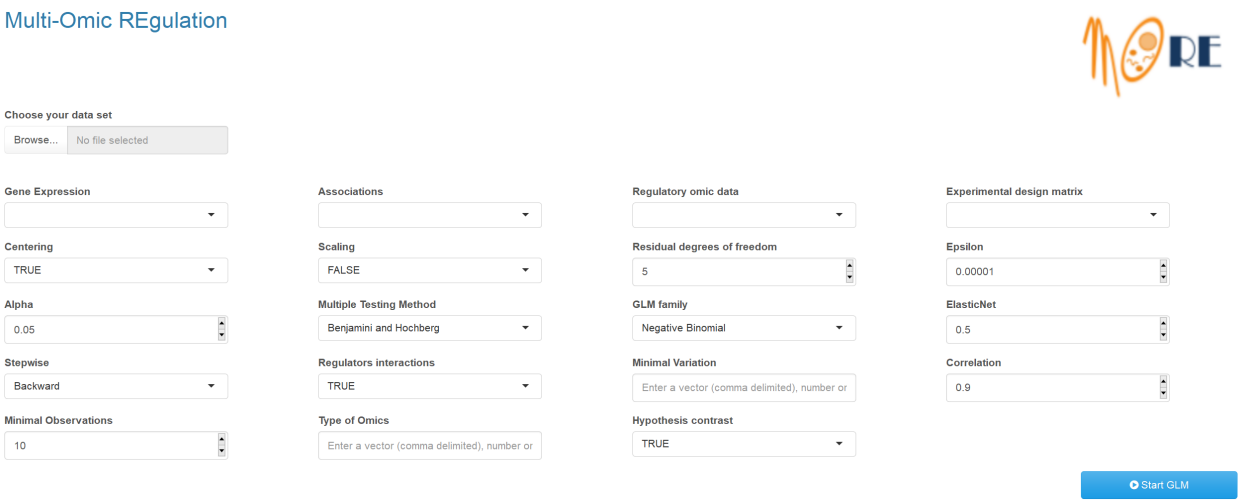


Figure 2: MORE Shiny

Gene expression matrix Matrix or data frame that contains expression values for each gene, in rows, under each experimental condition or replicate, in columns.

Experimental design matrix Matrix or data frame that contains the experimental covariates, such as treatments, points of time...

Regulatory omic data List that contains matrices or data frames containing the data for each regulatory omic, e.g. miRNA expression. The data frames structure is similar to gene expression data.

Association matrix List that contains data frames with the potential regulators for each regulatory omic considered. The association objects must be data frames and stored in a single list.

In this case, the file `TestDataShiny.RData` contains the objects described in **section 4.3**, unlike the experimental design matrix, that contains only one column with two conditions. By clicking on the button `Browse...`, users can choose their own data file (see Figure 3 blue box). Once the data is loaded, the user must enter the same input parameters of the GetGLM and RegulationPerConditon functions as defined in **section 4.1** and **section 5.1**.

It should be taken into account that if users want to enter a NULL value for a given parameter, they must leave the box empty.

In the example of the application, we will consider the input parameters shown in the following figure, leaving blank those we want to be NULL.

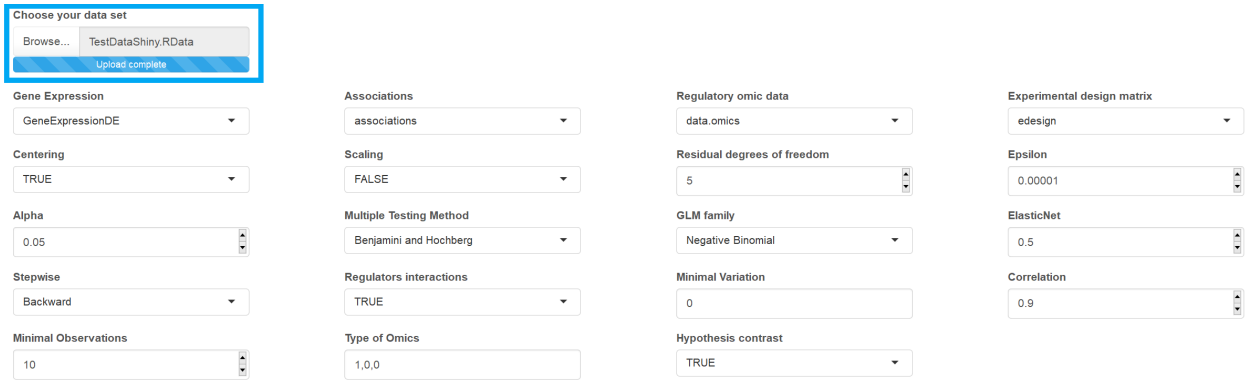


Figure 3: MORE Shiny: inputs for running example

Now, clicking the button `Start GLM`, we will obtain a summary table (see Figure 4). Specifically, this is the table defined in the **section 5.2**, the output of the **RegulationPerCondition()** function. The user can download the table in csv format by pressing the button `Download` (see Figure 4 orange box). It is necessary to save the file that will contain the table with name and extensin **.csv**.

The button `MORE plots` (see Figure 4) will generate plots to visualize the relationship between genes and regulators. The user can change the different parameters without re-executing the application to tune the plots or plot new elements.

Here we show the example for the gene ENSMUSG00000000078 (orange) and TF Mef2d (blue). Pressing the button `Generate Plot`, Shiny generates the first graph. However, if it is expected to obtain more than one graphic, the user can see all of them in a pdf file. This pdf file is generated by pressing the button `Download` (box orange in Figure 5) and saving the document, for example, as plotsGLM.pdf. It is essential to save the file with name and extension **.pdf**.

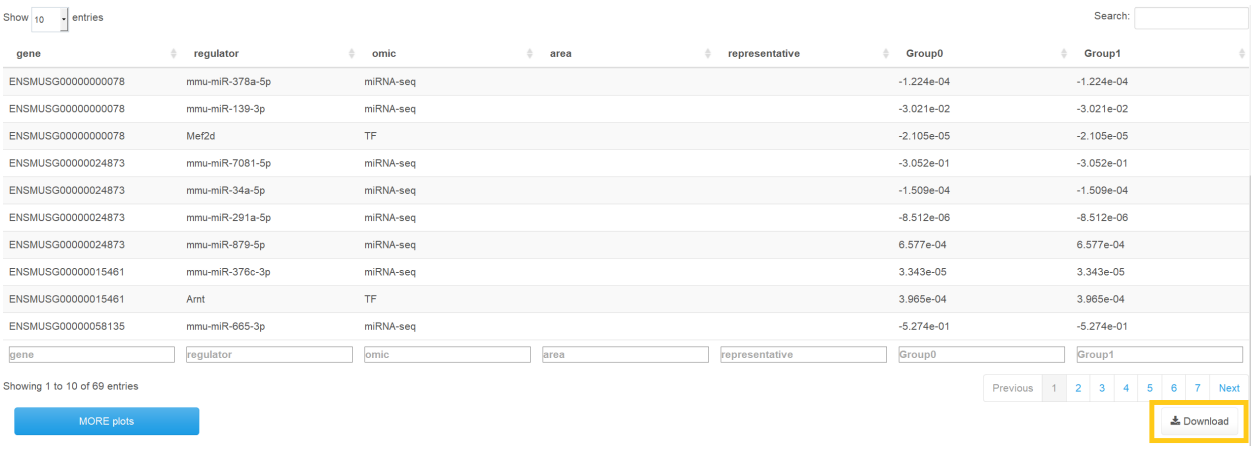


Figure 4: Summary table and `MORE plots` button. The user can download the whole table by pressing the button `Download` (orange box)

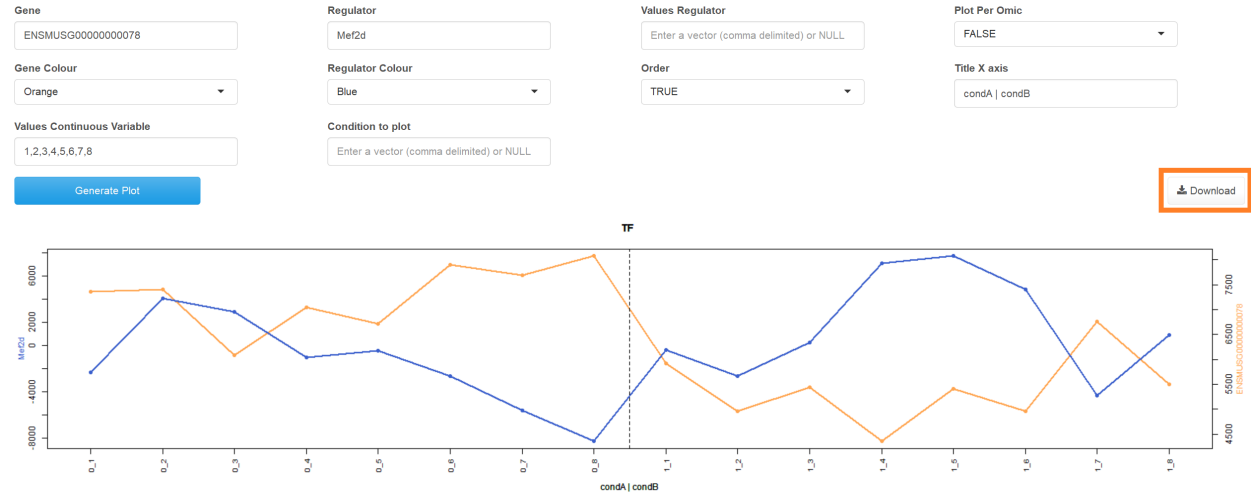


Figure 5: Inputs and plot for the pair gen–regulator (ENSMUSG00000000078–Mef2d). The user can download all plots by pressing the button `Download` (orange box)

8 How to cite MORE package

Tarazona, S., Tomás-Riquelme, B., Martínez-Mira, C., Clemente-Císcar, M., Conesa, A. (2018). MORE: Multi-Omics REgulation by regression models. R package version 0.1.0.