MIPRIP – User Manual

User Manual version 1.0
November 2, 2015

MIPRIP is a software package for R (www.r-project.org) to predict regulators of a gene of interest from gene expression profiles of the samples under study and known regulator binding information (e.g. from ChIP-seq/ChIP-chip databases). It is developed to study the specific regulation of the gene of interest in one group of samples compared to a control group. MIPRIP can straightforwardly be applied to similar problems integrating gene regulator binding information and expression profiles of samples of e.g. two different phenotypes, disease/healthy controls or treatment/controls.
For more details on this method, please see Poos *et al.* (2015).

MIPRIP author:
Alexandra Poos

Availability:
http://www.leibniz-hki.de/en/miprip.html

Publications:

Poos, A., Maicher, A., Dieckmann, A., Oswald, M., Eils, R., Kupiec, M., Luke, B. and König, R. (2015) Mixed Integer Linear Programming based machine learning approach identifies *regulators* of telomerase in yeast, *Nucleic Acids Research, submitted*.

Manual author:
Alexandra Poos

1. Installation

1.1. Required software

Before installing and running MIPRIP, the following software must be installed:

a) **R** (version 3.1.2); available from http://mirrors.softliste.de/cran/
b) **RStudio** (version 0.98.1103); available from https://support.rstudio.com/hc/en-us/articles/206569407-Older-Versions-of-RStudio-Desktop
c) **Gurobi** (version 6.04); available from http://www.gurobi.com/downloads/download-center

When running MIPRIP on Mac OS X, we suggest to use exactly the indicated versions of R, RStudio and Gurobi.

For the use of Gurobi a licence is necessary. For academic usage this licence is free and can be obtained after registration from http://www.gurobi.com/downloads/download-center (for more details see Quick Start Guide (http://www.gurobi.com/documentation/)). After installation and activation (grbgetkey) of the Gurobi software, the Gurobi R application programming interface (API) has to be installed with the Gurobi R package. You should do this with Rstudio by Tools – Install Packages – Install from Package Archive File. In case of a win64 installation, as Package archive you should write "C:/gurobi604/win64/R/gurobi_6.0-4.zip" (otherwise adjust to win32) or for a Mac OS X installation "/Library/Frameworks/R.framework/Versions/3.1/Resources/library". In addition, you need to install the package "slam" by the R command "install.packages("slam")".

1.2. Installation of MIPRIP

To install MIPRIP, download the package from http://www.leibniz-hki.de/en/miprip.html.

On a Unix/Linux system, execute the following command from a shell

```
R CMD INSTALL MIPRIP_1.0.tar.gz
```

or from the R command line

```
install.packages("MIPRIP_1.0.tar.gz")
```

or you can install it directly from RStudio similar to the Gurobi R API (click on Tools – Install Packages – Install from Package Archive File and select the file MIPRIP_1.0.tar.gz).

1.3. Loading MIPRIP

To load the package within the R command line simply type:

```
library(MIPRIP)
```

2. Running MIPRIP

This section explains how to run the MIPRIP algorithm with the available example data on the website.

2.1. Preprocessed data provided with the package

- Edge strength Matrix for *S. cerevisiae*
- Z-transformed gene expression data of 269 yeast deletion strains from Reimand *et al.* (1)
- Activity matrix for the regulators calculated from the gene expression data above
- Phenotypic class labels (telomere length of the 269 deletion strains, taken from Askree *et al.* (2004), Gatbonton *et al.* (2006), Ungar *et al.* (2009), Shachar *et al.* (2008) and Ben-Shitrit *et al.* (2012) (2-6))

2.2. Workflow of MIPRIP described on the example of our case study

A schematic overview of the workflow is given in Figure 1 based on the example of our case study. To predict the transcript levels of *EST1* in knockout strains showing short telomeres (short *tlms*) compared to control samples, we divided the data into a short *tlm* (18 samples) and a control dataset (241 samples). With both datasets, we performed a ten-times sixfold cross validation. Explicitly, the algorithm proceeded as follows:
(i) We randomly selected 120 datasets of all control samples (drawing with replacement).
(ii) Datasets of the 18 short *tlms* mutants as well as of the 120 control samples were randomly divided into six equally sized partitions. Five sixths (15 short *tlm* samples, 100 controls) were used to train the regulatory model and the remaining samples were used to validate the predictions.
(iii) The modelling was done separately for the short *tlm* mutants and the control datasets. To obtain solutions of a large range of model sizes, we generated models of different sizes:

(a) we started with constraining the models to use only one regulator, i.e. the number of β-parameters was limited to a maximum of two (β_0 and one β for the optimal regulator).

(b) This was repeated increasing the limit by one, allowing more β-parameters to be used by the model.

(c) Step (b) was repeated until the allowed number of β-parameters reached eleven (for ten regulators with known binding information to the target gene, plus β_0).

For each number of regulators we further did a fivefold inner cross-validation, which means the training dataset was divided into fifths and four fifths were used to determine the combination of regulators (training phase of the model) and the remaining one fifth to test the performance of the combination. The combination with the best performance was then used for the remaining steps.
Running (a) to (c) yielded two lists of selected regulators, one list for the short *tlm* mutants, and one list for the controls.
(iv) Steps (ii) and (iii) were repeated six times covering all possible partitions to be training sets.
(v) To estimate the performance of the models, we used the validation datasets and calculated the Pearson correlation of the predicted gene expression values (based on the

β-parameters yielded from the training sets) and the real expression values of the validation datasets from all six runs.
(vi) Steps (ii) to (v) were one cross validation, and we repeated these steps ten times.
(vii) For the control samples, steps (i) to (vi) were repeated ten times to cover their large variety.
For each *EST* gene, altogether 60 different models were constructed for the short *tlm* mutants and 600 for the control samples. For each model, we counted how often each regulator was selected by the optimizer. With these distributions we did a one-sided Wilcoxon Test for each regulator in the list to identify significantly different regulators between the short *tlm* mutants and the control samples. Significant levels (p-values) were corrected for multiple testing using the Benjamini-Hochberg method (7).

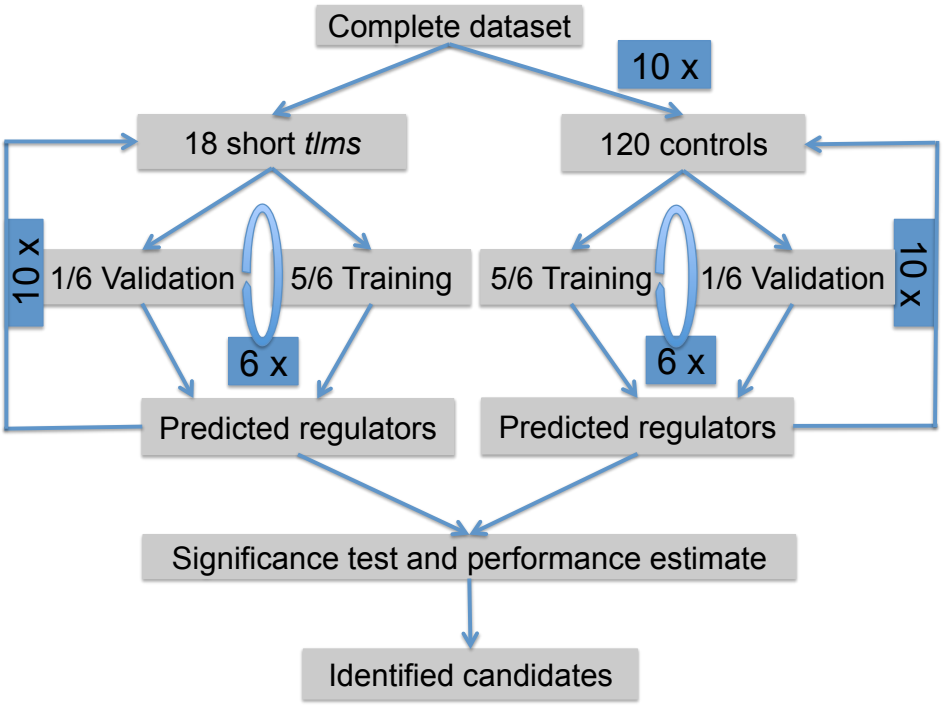

Figure 1. Schematic overview of the computational workflow. Using the expression profiles of short *tlm* knockout mutants as well as randomly selected control samples, a cross-validation was performed employing our linear modeling approach (inner loop for parameter optimization not shown). A significance test was performed to identify significant regulators being highly relevant for explaining expression of short *tlm* knockouts, while not being relevant for the controls.

To run MIPRIP, use the following R command:

```
miprip.result <- miprip.run(group1=<class1>, group2 = <class2, "Not defined">,
                num_repeats, num_cv, num_parameter, inner_cv =FALSE/TRUE,
                num_cv_inner, sample_size_group1 = length(group1),
                sample_size_group2 = length(group2), further_repeats_group1 = 1,
                further_repeats_group2 = 1)
```

Remark: MIPRIP was developed to compare regulation patterns between two different groups, e.g. different phenotypes, disease/healthy controls or treatment/controls. But it can also be used to study the regulation of a specific target gene in only one group.

In the following, the mandatory as well as the optional parameters are described in more detail.

2.2.1. Mandatory parameters to miprip.run()

The following parameters MUST be specified within the function miprip.run:

- group1=<class1>: vector of samples belonging to class1; please be aware that the the sample names in the classification file and in the gene expression dataset are equal; the best is to select only from [A-Z][a-z][0-9] for the sample names. Be careful with special characters!
- num_repeats: number of repeats; how often the cross-validation (defined by num_cv) should be repeated
- num_cv: specification of cross-validation runs, e.g. num_cv=3 for a three-fold cross-validation
- num_parameter: number of maximal parameters used for the modelling

2.2.2. Optional parameters to miprip.run()

The following parameters are optional. If no value is defined within the function call, the default values are used.

- group2=<class2, "Not defined">: specification if a second group as reference/control is used for the modelling; The **default is "Not defined"**; otherwise class2 should be a vector of samples belonging to class2.
- inner_cv=FALSE/TRUE: specifies if a further inner cross-validation should be performed to optimize the performance of the model (in our case study we get an overall higher performance of 0.1 for each target gene using an inner cross-validation). The **default is FALSE**.
- num_cv_inner: number of inner cross-validation, only if inner_cv=TRUE
- sample_size_group1: number of samples of group1 used for the model. The default is that all samples of group1 are used (length(group1)). Otherwise the number of selected samples is randomly chosen from all samples of group1.
- sample_size_group2: number of samples of group2 used for the model. The default is that all samples of group2 are used (length(group2)). Otherwise the number of selected samples is randomly chosen from all samples of group2.
- further_repeats_group1: specifies how often the modelling should be repeated with another randomly selected dataset to cover the variety, when not all samples of the corresponding group were used for the modelling, e.g. when only 120 out of 241 control samples were selected as in our case study. The **default is 1**, which means no further repeats.
- further_repeats_group2: specifies how often the modelling should be repeated with another randomly selected dataset to cover the variety, when not all samples of the corresponding group were used for the modelling, e.g. when only 120 out of 241 control samples were selected as in our case study. The **default is 1**, which means no further repeats.

See the help function in R for further information: ?miprip.run

2.2.3 Other functions included in miprip.run

There are two further functions included in the MIPRIP package, which are run by the miprip.run function:

- **without_inner.run**: is used for the modelling when no inner cross-validation is performed (details see ?without_inner.run)
- **inner.run**: is used for the modelling when a further inner cross-validation for performance optimization is performed (for details see ?inner.run)

The parameters of these two functions are defined through the miprip.run function.

3. Output of MIPRIP

The results of MIPRIP with all details are saved in the file "MIPRIP_results_<target_gene>.RData". To have a look at the results, load the file into an R session:

```
result=load("MIPRIP_results_<target_gene>.RData")
```

3.1. Using only one group:

- results_complete_group1: results of group1 for all cross-validation runs and repeats. For each run, a table with the number of parameters, the chosen regulators, the beta values of these regulators plus beta zero as well as the performance (Pearson correlation between the predicted and the gene expression values in the dataset) of the model is listed
- predictions_group1: predicted value of each validation sample over all tested regulator combinations compared to the gene expression value of the dataset
- frequency_group1: shows how often each regulator was chosen by the model for all regulator combinations for all runs (see 4.2.1 example1$frequency)
- performance_group1: correlation between the predicted value and the gene expression value from the dataset for all cross-validation runs and repeats
- result_complete_inner_group1: results of the inner cross-validations

3.2. Using two groups:

The main output of MIPRIP using two groups is a table with significant regulators of group1 compared to group2. For this, a Wilcoxon Test with the regulator frequencies (how often the regulators were used over all combinations and all cross-validation runs) is performed. The table contains the p-value and the Benjamini-Hochberg corrected p-value for each regulator (for more details see section 2.2.).

4. Examples

In the following two examples based on our case study are described. All necessary data is available at our website and with that data the example can be easily replicated.

4.1. Running MIPRIP on the example data (step by step)

(1) Loading the packages: After opening R from the terminal or using RStudio, the packages "gurobi" as well as "MIPRIP" have to be loaded:

```
library(gurobi)
library(MIPRIP)
```

(2) Loading of the dataset (X) as well as the activity matrix (Act) and the edge strength matrix (ES) with all the transcription factor to target interactions.

```
X<-read.delim(file="Expression_zTrans.csv", header=TRUE, check.names=FALSE)
ES=read.delim(file="Edge_strength_matrix.csv", header=TRUE, check.names=FALSE)
Act<-read.delim(file="Activity.csv", header=TRUE, check.names=FALSE)
```

Remark: The activity matrix is calculated by $act_{tk} = \frac{\sum_{i=1}^{n} es_{ti} * |g_{ik}|}{\sum_{i=1}^{n} es_{ti}}$ (8), (9), where act_{tk} was the estimated effect of regulator t in strain k, es_{ti} the edge strength between regulator t and gene i, g_{ik} the gene expression of gene i in strain k. It is the cumulative effect of a regulator on all its target genes, normalized by the sum of all target genes to balance regulators with high and low numbers of targets. Instead of using the activity matrix it is also possible to use the gene expression of the regulator t. In this case restrict the gene expression matrix X to only the regulators.

(3) For a classification into groups:

```
classification=read.table(file="Sample_classes.txt", header=TRUE)
#class1=short tlms, class2=control, class3=long tlms

class1=rownames(classification)[classification[,1]==1]
class2=rownames(classification)[classification[,1]==2]
```

(4) Definition of the target gene, e.g.

```
target_gene = "EST1"
```

(5) Definition of the parameters used by Gurobi (for more details see http://www.gurobi.com/documentation/6.0/refman/parameters.html):

e.g. `params <- list(timeLimit = 600,OutputFlag=0)`, otherwise the default parameters of Gurobi are used

(6a) Running MIPRIP with only one group:

```
example1=miprip.run(group1=class1, num_repeats=2, num_cv=3, num_parameter=5)
```

(6b) Running MIPRIP with 2 groups:

```
example2=miprip.run(group1=class1, group2=class2, num_repeats=2, num_cv=3, num_parameter=5, sample_size_group2=120)
```

4.2. Example results:

In the following the results of our two examples are described.

4.2.1. Results Example1

The results over all runs are saved in the file "MIPRIP_results_EST1". For more description of the output parameters, see section 3.1.

```
> result_complete_group1
[[1]]
                TF
1              HST1
2          CST6,SUM1
3       GCN4,MSN4,SUM1
4    GCN4,IXR1,MSN4,SUM1
5 SFP1,SIN3,SUM1,TEC1,TUP1
                                                                Betas_TF
1                                              2.98220592604133;-2.25456112419853
2                                                  -1.24671240711483;4.33141129872688;-
2.19386593626936
3                                  -27.0142677269812;27.0989485320169;3.56674777379556;-
2.77410708244327
4                                                                  -33.4904849431723;-
15.8140291648918;50.1797925850437;2.67763410878037;-2.77587646113504
5                                                                                  -
37.887318579444;2.52958716327282;4.21606443175813;18.3845790580579;13.846
6603662951;-0.485151755088737
  Amount TF      correlation
1       1         0.938668952014549
2       2         0.639717948634648
3       3         0.701342112665767
4       4         0.652872429916327
5       5         0.944732313600001

…

> performance_group1
  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6
1  0.94  0.93  0.61  0.96  0.92  0.41
2  0.64  0.68  0.70  0.66  0.90 -0.82
3  0.70  0.23 -0.59  0.59  0.83  0.81
4  0.65  0.18  0.71  0.37  0.80  0.78
5  0.94  0.22  0.63  0.06  0.87  0.35

> frequency_group1
     [,1] [,2] [,3] [,4] [,5] [,6]
CBF1  "0"  "0"  "0"  "0"  "0"  "0"
CST6  "1"  "0"  "0"  "1"  "0"  "0"
CUP2  "0"  "0"  "0"  "0"  "1"  "0"
FHL1  "0"  "0"  "2"  "0"  "0"  "0"
```

```
GCN4  "2"  "0"  "0"  "1"  "0"  "3"
HSF1  "0"  "0"  "0"  "0"  "0"  "0"
HST1  "1"  "4"  "1"  "2"  "4"  "3"
IXR1  "1"  "2"  "0"  "1"  "1"  "0"
MBP1  "0"  "0"  "0"  "0"  "0"  "1"
MIG1  "0"  "0"  "0"  "1"  "0"  "0"
MSN4  "2"  "0"  "1"  "0"  "2"  "2"
RFX1  "0"  "1"  "1"  "0"  "0"  "1"
SFP1  "1"  "0"  "1"  "0"  "2"  "0"
SIN3  "1"  "0"  "1"  "2"  "0"  "1"
SIN4  "0"  "1"  "0"  "0"  "0"  "0"
SNF2  "0"  "1"  "1"  "0"  "0"  "0"
SNF6  "0"  "0"  "0"  "0"  "0"  "0"
SPT20 "0"  "0"  "1"  "1"  "2"  "0"
SRB2  "0"  "4"  "0"  "2"  "0"  "1"
STE12 "0"  "0"  "0"  "0"  "0"  "0"
SUM1  "4"  "1"  "4"  "3"  "1"  "2"
SWI3  "0"  "0"  "1"  "0"  "2"  "0"
SWI4  "0"  "0"  "0"  "0"  "0"  "0"
TEC1  "1"  "0"  "0"  "1"  "0"  "1"
TUP1  "1"  "1"  "1"  "0"  "0"  "0"
```

```
> predictions_group1
[[1]]
              1           2           3           4           5           g_real
HFI1  1.59910281  2.0271596  1.1863140 -0.3281128  0.00204397 -0.002215919
HST1  5.50390254  2.1653698  1.7361444  1.4822265  3.05341715  3.605533549
MOT2 -0.99446219 -0.8992339 -0.5103011  0.1760532  0.19861619 -0.721002871
PGD1 -0.02395076  0.5770387  0.2824315  0.4664661  0.22590610 -1.031517697
RSC2  0.50609210  0.3380023  0.4139242  0.7547855 -0.08888033 -0.224535023
SRB5 -1.02695535 -0.8906986 -0.9872743 -0.7315673 -0.20329960 -0.284645710
```

…

4.2.2. Results Example2

The results of both groups over all runs are saved in the file “MIPRIP_results_EST1”. For more description of the output parameters, see section 3.2.
The significant regulators of group1 compared to group2 are given out directly:

```
Significant Hits of target gene EST1
      TF              p-value                  p-value_BH
[1,]  "HST1"  "0.00283952053929198"  "0.0354940067411497"
[2,]  "SUM1"  "0.00283952053929198"  "0.0354940067411497"
[3,]  "SIN3"  "0.013442755926704"    "0.112022966055867"
[4,]  "RFX1"  "0.0352664245674461"   "0.132160660689705"
[5,]  "MSN4"  "0.0355106009184896"   "0.132160660689705"
[6,]  "GCN4"  "0.0370049849931174"   "0.132160660689705"
[7,]  "SRB2"  "0.0370049849931174"   "0.132160660689705"
[8,]  "CUP2"  "0.202328380963643"    "0.562023280454564"
[9,]  "FHL1"  "0.202328380963643"    "0.562023280454564"
```

[10,]	"SWI3"	"0.263544628432769"	"0.619575544010488"
[11,]	"CST6"	"0.297396261125034"	"0.619575544010488"
[12,]	"SNF2"	"0.297396261125034"	"0.619575544010488"
[13,]	"TUP1"	"0.427566070292353"	"0.82224244286991"
[14,]	"TEC1"	"0.536702014145331"	"0.91530885577001"
[15,]	"MIG1"	"0.549185313462006"	"0.91530885577001"
[16,]	"SIN4"	"0.597229705666253"	"0.93317141510352"
[17,]	"MBP1"	"0.800462401725881"	"1"
[18,]	"SFP1"	"0.801756425501317"	"1"
[19,]	"SPT20"	"0.933929735116623"	"1"
[20,]	"CBF1"	"0.977072972007091"	"1"
[21,]	"HSF1"	"0.991520748769403"	"1"
[22,]	"IXR1"	"0.999298776375666"	"1"
[23,]	"SNF6"	"0.999298776375666"	"1"
[24,]	"STE12"	"1"	"1"
[25,]	"SWI4"	"1"	"1"

Remark: Compared to our case study as described in Poos *et al.* we run this example without an inner cross-validation, with less repeats and less number of parameters in order to decrease running time. But our most promising hits, Sum1 and Hst1, were again top selected with this parameter setting.

Acknowledgements:

We thank Volker Ast, Marcus Oswald and Theresa Schacht for their support and testing and Joao Saraiva for proof-reading as well as setting up the website.

References:

1. Reimand, J., Vaquerizas, J.M., Todd, A.E., Vilo, J. and Luscombe, N.M. (2010) Comprehensive reanalysis of transcription factor knockout expression data in Saccharomyces cerevisiae reveals many new targets. *Nucleic acids research*, **38**, 4768-4777.
2. Askree, S.H., Yehuda, T., Smolikov, S., Gurevich, R., Hawk, J., Coker, C., Krauskopf, A., Kupiec, M. and McEachern, M.J. (2004) A genome-wide screen for Saccharomyces cerevisiae deletion mutants that affect telomere length. *Proceedings of the National Academy of Sciences of the United States of America*, **101**, 8658-8663.
3. Ben-Shitrit, T., Yosef, N., Shemesh, K., Sharan, R., Ruppin, E. and Kupiec, M. (2012) Systematic identification of gene annotation errors in the widely used yeast mutation collections. *Nature methods*, **9**, 373-378.
4. Gatbonton, T., Imbesi, M., Nelson, M., Akey, J.M., Ruderfer, D.M., Kruglyak, L., Simon, J.A. and Bedalov, A. (2006) Telomere length as a quantitative trait: genome-wide survey and genetic mapping of telomere length-control genes in yeast. *PLoS Genet*, **2**, e35.
5. Shachar, R., Ungar, L., Kupiec, M., Ruppin, E. and Sharan, R. (2008) A systems-level approach to mapping the telomere length maintenance gene circuitry. *Molecular systems biology*, **4**, 172.

6. Ungar, L., Yosef, N., Sela, Y., Sharan, R., Ruppin, E. and Kupiec, M. (2009) A genome-wide screen for essential yeast genes that affect telomere length maintenance. *Nucleic acids research*, **37**, 3840-3849.
7. Benjamini, Y. and Hochberg, Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J Roy Statist Soc Ser B*, **57**, 289-300.
8. Schacht, T., Oswald, M., Eils, R., Eichmuller, S.B. and Konig, R. (2014) Estimating the activity of transcription factors by the effect on their target genes. *Bioinformatics*, **30**, i401-407.
9. Poos, A., Maicher, A., Dieckmann, A., Oswald, M., Eils, R., Kupiec, M., Luke, B. and König, R. (2015) Mixed Integer Linear Programming based machine learning approach identifies *regulators* of telomerase in yeast, *Nucleic Acids Research, submitted*.