ReL package manual

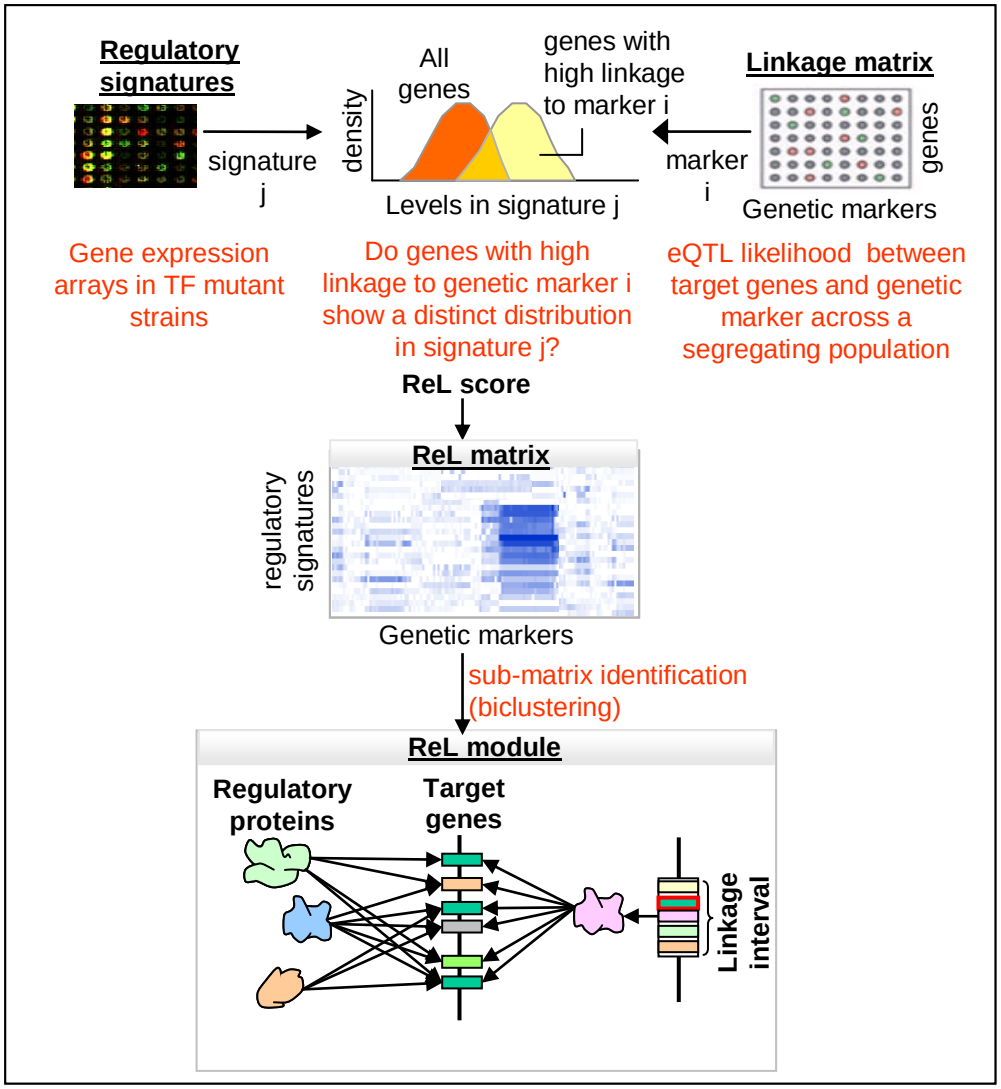
Understanding gene sequence variation in the context of transcription regulation in yeast

Irit Gat-Viks, Renana Meller, Martin Kupiec, Ron Shamir

1. Introduction

The following manual describes all scripts and executable required for the creation of ReL modules. We advise the starting user to read only the Quick Start tutorial (Section 2), which provides instructions on how to apply the entire flow automatically. Sections 3-7 should be read only for a more sophisticated usage. Sections 3-4 provides detailed information on variables and input formats. Section 5 provides details on how to create a ReL matrix. The biclustering tool is then detailed in Section 6, and the final formatting of the output appears in Section 7.

The package runs on 64 bit Linux environment and requires JRE version 1.5.0_05 or higher and c library libc.so.6 with Glibc 2.4 or higher.



2. Quick start

The following section describes how to create ReL modules using default parameters. Sections 3-7 should be used in a case of a more sophisticated usage.

The c shell script RunFlow.csh runs the entire flow automatically. The input is the compendium file, linkage matrix, and several organism-specific files (see below). The output is a collection of ReL modules, each consists of a set of regulatory proteins, a set of target genes, and a single linkage interval.

Step1: Download ReLPackage.zip and extract it. Make sure that you have rwx permissions for the entire ReLPackage directory.

Step 2: Edit RunFlow.csh: The variable REL_PACKAGE_ROOT should be set to the full path in which you have extracted the zip file.
For example:

```
## Update the root directory and required environment variable.
setenv REL_PACKAGE_ROOT "<YOUR_HOME_FOLDER>/ReLPackage"
```

Step 3: Run RunFlow.csh as follows:

```
> RunFlow.csh <Input compendium file> <Organism directory> <Output
file> <Log file>
```

Input compendium file – the compendium of regulatory signatures.
Organism directory – the directory that includes the linkage matrix and all other organism-specific files.
Output file – the output file (a collection of ReL modules).
A Log file

Input and output files specification appears in sections 4 and 5.1. An example of input files for the yeast system can be found in the Organism/Yeast directory and in the Compendium/RegulatorySignatures.txt file. An example output file can be found in Output/ReLModule.xls. The complete output file could be generated as follows:

```
> RunFlow.csh Compendium/RegulatorySignatures.txt Organism/Yeast
Output/ReLModules.txt RunFlow.log
```

Additional comments:
- For the yeast's linkage matrix and 100 signatures, the ReL package calculates modules within 3-4 hours on a standard linux machine.
- Intermediate files are created in the Output directory.

3. Environment variables

In order to run the Perl script you should define REL_PACKAGE_ROOT environment variable. The REL_PACKAGE_ROOT should be the full path in which you uncompressed ReLPackage.zip.

```
> setenv REL_PACKAGE_ROOT "<YOUR_HOME_FOLDER>/ReLPackage"
```

In order to run MatchMarkerToSignature executable, the LD_LIBRARY_PATH should be set as follows:

```
> setenv LD_LIBRARY_PATH
"<YOUR_HOME_FOLDER>/ReLPackage/MatchMarkerToSignature"
```

4. Organism designated files

Four organism-specific files are required: LinkageMatrix.txt, GenomicFeatures.txt, GeneticMarkersDictionary.txt and ProteinDifferences.txt. Yeast files are available in the Organism/Yeast directory. If you wish to work with a different organism, create your own directory under the Organism folder and prepare four input files with the same names within your directory. Importantly, apply the dos2unix command on all input files.

4.1 Linkage matrix file

The linkage matrix contains eQTL likelihood score for each pair of gene and genetic marker.
There are several requirements for the linkage matrix file:

- The file should be tab-delimited file
- The rows should be genetic marker ids and the columns should be genes. If this is not the case use the Transpose.pl script found in the PerlScripts directory to transpose the file.
  `> perl Transpose.pl <Input file> <output file>`
- The genetic markers ids are sequential numbers that matches the ids as appear in GeneticMarkersDictionary.txt.
- Missing or unknown values should have the value -1000. If you want to use a different value as missing value you should update the linkageMatrixMissingData parameter in the option file (see section 5.2.2).

4.2 Genomic features file

The genomic feature file is a tab delimited file that contains data about features along the genome. The genomic feature file may contain several feature types, however only "ORF" and "telomere" features will be read from the file. The required structure of the file:

Column #	Data
1	feature type (we read only "ORF" and "telomere" features)
2	empirical evidence (e.g., Dubious, Uncharacterized, Verified)
3	gene name (e.g., ORF name in yeast)
4	alias name 1
5	alias name 2
8	chromosome
9	start position
10	end position
11	C/W strand
15	description

(Column # starts from 0). All the other columns will be ignored.

4.3 Genetic markers dictionary

The genetic marker dictionary is a tab delimited file that matches each genetic marker ids to a genomic locus. Missing or unknown values should have the value "none". If

you want to use a different value as missing value, update the missingValue parameter in the option file (see section 5.2.3). The genetic marker dictionary has a header line and its required structure is:

Column #	Data
0	marker id (sequential number)
2	ORF name
3	chromosome
4	genomic position

All the other columns will be ignored.

4.4 Protein differences file (optional)

The protein differences file is a tab-delimited file, which contains the protein difference between the two parental strains. The required structure of the file:

Column #	Data
0	gene name of the first strain (the same gene name as in GenomicFeatures.txt)
1	the gene name of the ortholog protein in the second strain
2	a textual description of the differences between the proteins

All the other columns will be ignored.

The protein differences file is not mandatory. Instead, it is possible to create an empty ProteinDifferences.txt file.

5. Creating ReL matrix

To create the ReL matrix, the user should provide input files and then run the MatchMarkerToSignature executable.

5.1 Signatures compendium input file

A tab-delimited file of –log (intensity ratio). Rows are regulatory signatures and columns are genes. Missing or unknown values should get the value -1000. If you want to use a different value as missing value you should update the signaturesMissingData parameter in the option file (see section 5.2.2).

An example can be found in Compendium/RegulatorySignatures.txt

5.2 Matching genetic marker to signature

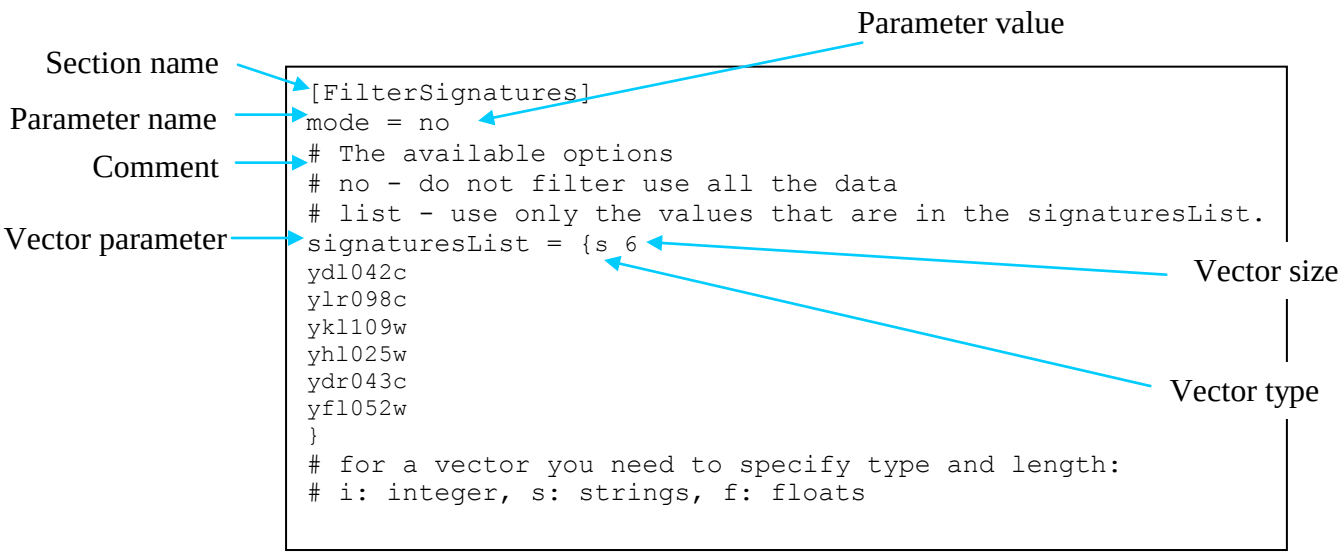
The MatchMarkerToSignature executable should be used as follows:

```
> MatchMarkerToSignature <Option file>
```

For example:

```
> MatchMarkerToSignature/bin/MatchMarkerToSignature
MatchMarkerToSignature/MatchMarkerToSignatureOptions.txt
```

The parameters in the option file are divided to sections. A parameter might be a scalar or a vector of type s (string), i (integer), or f (float).



5.2.1 Score section

The Score Section contains the following parameters:

type

The main score type. We currently support only HG (the hyper geometric test). Two additional parameters should be defined when using the HG test: signatureThresholdType and markerLinkageThresholdType.

In order to obtain a ReL score, the AdditionalTests vector should include the t-test score (see below).

signatureThresholdType
The threshold that divides the regulatory signature data into up-regulated and down-regulated genes. The available options for this parameter are:

- **0** - FIX - the threshold is determined by the user. When using this option, the signatureThreshold parameter must be defined.
- **1** - MEAN - the threshold is the mean of all the values.

markerLinkageThresholdType
The eQTL likelihood threshold. The available options for this parameter are:

- **4** - ALL - check all the values as thresholds. The chosen threshold is the one that maximizes the HG score
- **5** - PERCENTAGE – takes the top X values and examines them as possible thresholds. The chosen threshold is the one that maximize the HG score. When using this option, the markerLinkageThresholdPercentage parameter must be defined.
- **6** - FIX_PERCENTAGE – a fixed threshold that is determined by the user. When using this option, the markerLinkageThresholdPercentage parameter must be defined.

AdditionalTests
A vector for additional tests to be printed in the output file. The available tests are:

- **TTest** – a t-test comparing the population below the eQTL likelihood threshold and above eQTL likelihood threshold. When applied in conjugation with the HG score (type = HG), we call it the ReL score.
- **GroupSizeTest** – the number of hit genes.
- **AboveThresholdGroupSizeTest** – the number of high-linkage genes.
- **LODScoreThresholdTest** – the value of the chosen eQTL likelihood threshold.

In order to perform filtering in later stages, the GroupSizeTest and TTest options must appear in the AdditionalTests vector (see Section 6.1).

TestLogic
Filtering results. The package currently supports only one option: the aboveThresholdsLogic.

- **aboveThresholdsLogic** – filter only scores that are above the specified threshold. When using this option, the TestThresholds vector must be defined. In the following example, we require that the HG score will be greater than -1, the GroupSizeTest score will be greater than 0 and the TTest score will be greater than -1. By using these thresholds, we do not filter any result.

```
[Score]
type = HG
# The available methods of linkage analysis are:
# HG - the hypergeometric test
# (and then you must define signatureThresholdType and
# markerLinkageThresholdType)
signatureThresholdType = 1
# the available threshold type are:
# 0 - FIX - the threshold is determined by the user, must define
# signatureThreshold value.
# 1 - MEAN - the threshold is the mean of all the values.
markerLinkageThresholdType = 5
markerLinkageThresholdPercentage = 0.01
# the available threshold type are:
# 4 - ALL - check all the values as thresholds.
# 5 - PERCENTAGE - check a percentage with the highest values as a
# threshold, must define markerLinkageThresholdPercentage.
# 6 - FIX_PERCENTAGE - the threshold is determined by the user,
# must define markerLinkageThresholdPercentage.

AdditionalTests = {s 2
GroupSizeTest
TTest
}
# the available tests are:
# TTest
# GroupSizeTest
# AboveThresholdGroupSizeTest
# LODScoreThresholdTest

TestLogic = aboveThresholdsLogic
# the available logics are:
# aboveThresholdsLogic - check that all the tests return scores
# that are above the threshold, must define TestThresholds vector.
TestThresholds = {f 3
-1
0
-1
}
```

5.2.2 Data section

The Data section contains the following parameters:

signaturesCompendium
The signature compendium input file.

signaturesCompendiumMissingData
A numeric value which indicates missing values and unknown values in the signatures compendium input file.

linkageMatrix
The linkage matrix file.

linkageMatrixMissingData
The numeric value that indicates missing and unknown values in the linkage matrix.

```
[Data]
# The signatures compendium file should answer the following
# requirements:
# 1. The missing data should be according to the
#    signaturesCompendiumMissingData.
# 2. The file should be converted to Unix format.
# 3. The rows should be regulatory signatures and the columns
#    should be genes. If this is not the case use the script
#    PerlScripts/Transpose.pl.

signaturesCompendium =
REL_PACKAGE_ROOT/Compendium/RegulatorySignatures.txt
signaturesCompendiumMissingData= -1000
linkageMatrix = REL_PACKAGE_ROOT/Organism/Yeast/LinkageMatrix.txt
linkageMatrixMissingData = -1000
```

5.2.3 GeneticMarkerDictionary Section.

dictionaryFile
The GeneticMarkerDictionary.txt file.

keyColumn
The 'marker id' column in the dictionary file.

valueColumn
The column indicating 'ORF name' in the dictionary file.

missingValue
The parameter describes how missing and unknown values are indicated in the dictionary.

```
[GeneticMarkerDictionary]
dictionaryFile =
REL_PACKAGE_ROOT/Organism/Yeast/GeneticMarkersDictionary.txt
keyColumn = 0
valueColumn = 2
missingValue = none
```

The format of GeneticMarkersDictionary.txt is detailed in Section 4.3.

5.2.4 FilterGeneticMarkers section

The FilterGeneticMarkers section contains the following parameters:

mode
Indicates whether and how genetic markers should be filtered. The available options for this parameter are:

- **no** – do not filter; use all genetic markers.
- **all** - use all genetic markers ids that appear in the GeneticMarkerDictionary
- **list** - use only the marker ORF names that are in the geneticMarkersList vector. When using this option you must define geneticMarkersList vector.

- **intervals** - use only the intervals of genetic marker ids that appears in the geneticMarkersList.When using this option you must define geneticMarkersList vector.

geneticMarkersList
A list of genetic markers as indicated in the mode parameter.

```
[FilterGeneticMarkers]
mode = intervals
# The available options
# no - do not filter use all the data.
# all - use only genetic markers that appears in the
# GeneticMarkerDictionary
# list - use only  marker ORF names that are in the
# geneticMarkersList, example: NHR001C.
# intervals - use only the intervals of genetic marker ids that are
# in the geneticMarkersList, example: 1459-1501.

geneticMarkersList = {s 2
1231-1260
353-379
}
```

5.2.5 FilterSignatures section
The FilterSignatures section contains the following parameters:

mode
Indicates whether and how the signatures should be filtered. The available options for this parameter are:

- **no** – do not filter; use all signatures.
- **list** - use only the signatures that are in the signaturesList vector. When using this option you must define signaturesList vector.

```
[FilterSignatures]
mode = no
# The available options
# no - do not filter use all the data
# list - use only the values that are in the signaturesList.
signaturesList = {s 2
ste12Vswt
ylr451w
}
```

5.2.6 PostAction section
The PostAction section contains the following parameters:

action
Determines which additional files were created in addition to the output file. The available options for this parameter are:

- **no** – no additional files will be created.

- **reportEnrichedGroup** – create a file that for each pair of genetic marker and signature, lists all the hit genes. When using this option you must define enrichedGroupOutput parameter.
- **reportGroupAboveHGThreshold** – create a file that for each pair of genetic marker and signature, lists all the high-linkage genes. When using this option you must define enrichedGroupOutput parameter.

enrichedGroupOutput
File that contains list of genes as indicated in the action parameter.

```
[PostAction]
action = reportEnrichedGroup
# The available post actions are:
# no - do not perform post action.
# reportEnrichedGroup - report the hit genes in an output file.
# Must define enrichedGroupOutput parameter.
# reportGroupAboveHGThreshold - report the high-linkage genes in an
# output file. Must define enrichedGroupOutput parameter.

enrichedGroupOutput =
REL_PACKAGE_ROOT/Output/RegulatorySignatures.EnrichedGroupReport.txt
```

Important - use the reportEnrichedGroup option if you like to complete the analysis and create ReL modules (see section 7.1).

5.2.7 Out section
The Out section contains the following parameters:

outputfile
the location of the output file.

```
[Out]
outputfile =
REL_PACKAGE_ROOT/Output/RegulatorySignatures.MatchMarkerToSignature.txt
```

5.2.8 Match marker to signature output file
The MatchMarkerToSignature output file is a tab-delimited file. For each pair of genetic marker and signature, the file provides several scores as follows: The first score is always the hyper-geometric score. Additional scores are specified in the AdditionalTests vector parameter (see section 5.2.1) and appear at the same order. For example, if the AdditionalTests vector will be assigned with two values: GroupSizeTest and TTest, the output file will contain three scores for each pair of genetic marker and signature. The first is the HG score, the second is GroupSizeTest score, and the third score is the TTest score (ReL score). The ReL score is required to create the ReL matrix. The additional scores could be used to filter the ReL score (see section 6.1 for details).

6. Biclustering the ReL matrix

Here we provide details on how to bicluster the ReL matrix. Section 6.1 shows how to extract the ReL matrix from the output of MatchMarkerToSignature executable. Section 6.2. shows how to create biclustering seeds. Finally, Section 6.3 explains how to apply the biclustering algorithm.

6.1 Extract ReL score

To extract the ReL matrix, use the SplitScore.pl script.

Usage:

```
> perl SplitScoreFile.pl <Scores input file> <Score element location>
<Output file> [-filter "filter string"]
```

Score input file - output file of MatchMarkerToSignature.
Score element location - which element of the score to extract.
Output file - the name of the output file.
-filter - filter the Score input file according to a given filtering rule (see the following example).

Example:

```
perl PerlScripts/SplitScoreFile.pl
Output/RegulatorySignatures.MatchMarkerToSignature.txt 3
Output/RegulatorySignatures.ReLMatrix.txt -filter "2>=10"
```

Assume that the output of the MatchMarkerToSignature contains three scores: HG score, GroupSizeTest score (i.e., number of hit genes) and TTest score (i.e., the ReL score). In such case, the above command will extract the ReL score (Score element location = 3). A filtering rule "2>=x" indicates that the second score (i.e., number of hit genes) must be equal or greater than x. Using this filter, it is possible to enforce the biclustering algorithm to ignore ReL scores calculated based on a very small number of hit genes.

6.2 Creating seeds

To create set of seed ReL modules, use the CreateSeeds.pl script. This script takes as input a file of scores and outputs all marker intervals that are above a certain score threshold.

Usage:

```
> perl CreateSeeds.pl <Scores input file> <Threshold score> <Output
file>
```

Score input file – the ReL matrix
Threshold score - the seed threshold (as defined in **SText E**).
Output file - the name of the output file.

Example:

```
perl PerlScripts/CreateSeeds.pl
Output/RegulatorySignatures.ReLMatrix.txt 10
Output/RegulatorySignatures.Seeds.txt
```

The above command will produce seeds that contain all marker ids with at least one ReL score that is higher than the seed threshold 10.

6.3 ISA biclustering

To bicluster the ReL matrix, use the ISA.jar, which implements a modified ISA algorithm designed to fit the special requirements of ReL analysis.

Usage:

```
> java -jar ISA.jar <Scores input file> <Seeds file> <Threshold
score> <Output clusters number> <Output file>
```

Scores input file – the output of SplitScoreFile.pl script.
Seeds file – the output of the CreateSeeds.pl script.
Threshold score – the minimum ReL score of a module.
Output clusters number – the maximum number of ReL modules.
Output file - the name of the output file.

Example:

```
java -Xmx200m -jar ISA/ISA.jar
Output/RegulatorySignatures.ReLMatrix.txt
Output/RegulatorySignatures.Seeds 8 30
Output/RegulatorySignatures.Clusters
```

The above command will produce a file with at most 30 ReL modules. Each of these ReL modules should have a module ReL score that is greater than 8.

7. ReL modules output file

7.1 Generate the output file

The SignaturesTargetsAnalysis.pl and GenomeFeaturesAnalysis.pl scripts add additional information to the ReL modules file. In particular, the scripts add the following information:

1. Translate linkage intervals into genomic regions and add indication whether those regions are telomeric or not.
2. List all genes contained within the linkage interval. For each gene within the linkage interval, report all protein differences between the parental strains.
3. Calculate the set of target genes (calculated based on the sets of hit genes within the module, see Methods).
4. Mark cis and telomere target gene.

Usage:

```
> perl SignaturesTargetsAnalysis.pl <BiCluster file> <Enriched group
report file> <Genetic marker dictionary file> <Genome features file>
<Output file I>

> perl GenomeFeaturesAnalysis.pl <Output file I> <Genome features
file> <Protein difference file> <Output file II>
```

BiCluster file – the output file of ISA.jar
Enriched group report file - file that contains the hit genes for each pair of genetic marker and signature (see reportEnrichedGroup option in section 5.2.6).
Genetic marker dictionary file - file that contains the positions along the genome for each genetic marker (see Section 4.3).
Genome features file - file that contains features along the genome (see Section 4.2).
Output file I - the name of the SignaturesTargetsAnalysis.pl output file.
Protein difference file - file that contains the difference between the proteins of two yeast strains (see section 4.4)
Output file II - the ReL modules output file.

Example:

```
> perl PerlScripts/SignaturesTargetsAnalysis.pl
Output/RegulatorySignatures.Clusters.txt
Output/RegulatorySignatures.EnrichedGroupReport.txr
Organism/Yeast/GeneticMarkersDictionary.txt
Organism/Yeast/GenomicFeatures.txt
Output/RegulatorySignatures.SignaturesTargetsAnalysis.txt

> perl PerlScripts/GenomeFeaturesAnalysis.pl
Output/RegulatorySignatures.SignaturesTargetsAnalysis.txt
Organism/Yeast/GenomicFeatures.txt
Organism/Yeast/ProteinDifferences.txt Output/ReLModules.txt
```

7.2 output file format

The module file contains data about the ReL modules. The first column of the file is the module index. Each module contains the following sections:

linkage interval
Section that contains general information about the linkage interval. The linkage interval section contains the following columns:

1. **genetic locus interval**
2. **internal score**
3. **ReL score of the module**
4. **number of gentic marker in the module**
5. **number of signatures in the module**

The linkage interval section also contains the following lines:

1. **chromosome** – the chromosome that contains the linkage interval.
2. **start position** – the starting position of the linkage interval.
3. **end position** – the ending position of the linkage interval.
4. **telomeric region** – indication whether the interval is in a telomeric region.

features
Section that contain information about the genomic features along the linkage interval. The features section contains the following columns:

1. **gene name (e.g., yeast ORF name)**
2. **empirical evidence**
3. **alias name**
4. **alias name**
5. **chromosome**
6. **start position**
7. **end position**
8. **C/W strand**
9. **description**
10. **overlap target genes** – indication whether this feature appears in the target genes.
11. **matching protein** – matching protein in the RM11 strain.
12. **protein difference** - difference between the coding regions of the two parental strains BY and RM.
13. **close to telomere** - indication whether the feature is close to the telomere.

regulatory signatures
Section that contains the signatures of the module. The regulatory signature section contains the following columns:

1. **protein name (e.g., yeast ORF name)**
2. **average ReL score of the signature** – the signatures are ordered according to this score in descending order.
3. **cis indication** – indication whether this signature is considered cis signature. (Cis signatures will not effect the target genes section.)
4. **identity measure** – measure of the overlap between the group of hit genes of this signature and the main signature (first signature that is not cis-signature).

Values below 0.6 are indicated with "low". (Low identity measure signatures will not affect the target genes section.)

5. **empirical evidence**
6. **alias name**
7. **alias name**
8. **chromosome**
9. **start position**
10. **end position**
11. **C/W strand**
12. **description**
13. **close to telomere** - indication whether the signature is close to the telomere.

target genes

Section that contains the target genes of the module. The target genes section contains the following column:

1. **ORF name**
2. **appearance number** - number of times the target gene appears as a gene hit within the module sub-matrix (without cis or low identity signatures).
3. **empirical evidence**
4. **alias name**
5. **alias name**
6. **chromosome**
7. **start position**
8. **end position**
9. **C/W strand**
10. **description**
11. **close to telomere** – indication whether this target gene is close to the telomere.
12. **overlap features** - indication whether this target gene appears in linkage interval features.

To better illustrate the structure of the output file we provide excel spreadsheet that contains example of a ReL module with formatting and comments. The spreadsheet - ReLModule.xls - could be found in the Output directory. You can copy the conditional formatting using the excel paste special command to your output file. (First paste format to column A and then to columns B-N.)