# aiMeRA User's Guide

```
library(aiMeRA)
```

## Quick start

This is an illustration of how functions of the aiMeRA package can be used to uncover coupling in biological networks. A data set containing qPCR data is available from the package to provide an example. In addition, the data sets used in MRA original paper (Figures 7a and 7b, Kholodenko,et.al, PNAS, 2002) are also included.

We show how to compute the coupling between the Estrogen Receptor alpha (ER), the Retinoic Acid Receptor (RAR), and two corepressors of their transcriptional activity that are RIP140 (NRIP1) and LCoR. Data for three transfections (biological replicate) and two technical replicates per transfection are stored in six data tables called:

```
estr1_A     estr1_B    estr2_A    estrt2_B    estr3_A     estr3_B
```

The objective in this example is to retrieve the MRA connectivity coefficients (connectivity map) between the four modules mentioned above when the system is in a double-stimulation condition with estradiol (E2) and retinoic acid (RA).

### Reading the data

A specific format for the input data tables is required: the first column of each table must contain the names of the biological modules, and the columns names should be the names of the perturbations. In order to check whether the format of the data set is correct, we use the function `data.setup()`

```
data=data.setup(list(estr1_A,estr1_B,estr2_A,estr2_B,estr3_A,estr3_B))
```

This function also accepts a specific path to a folder containing the data table files and the perturbation rules file in a .cvs format

```
#Example of reading the data from a specific folder
data=data.setup(/User/some_path_to_folder_files/)
```

If more than one data table are checked, then a list of matrices in the correct format for the mra function is returned. If a single table is checked, then a single matrix in the right format is returned. It also checks for correspondance with the perturbation rules table.

### Perturbation rules table

**Perturbation rules** are a set of expressions to specify how perturbations are affecting biological modules. Since the aiMeRA package is based on classical Modular Response Analysis (MRA) it assumes that one perturbation must affect only one module of the network. This is specified as binary values in the **perturbation matrix**.

The perturbation matrix can be created from a set of perturbations rules using the function `read.rules()`

The rules syntax is `Perturbation->Module` for a single perturbation,`Perturbation->0` to specify the basal line and optionally `0->Module` to specify modules that were not perturbated, but for which connectivity issued from other modules exists (unidirectional MRA).

For this example we define the elements of our network as follows:

**Biological modules:**

- **LCoR** LCoR gene expression
- **RIP140** NRIP1 gene expression
- **Luciferase** ER transcriptional activity reporter
- **Hoxa5** RAR transcriptional activity reporter

**Perturbations:**

- **E2+RA+siLCoR:** Downregulation of LCoR by a siRNA in the double-stimulation condition.
- **E2+RA+siRIP140:** Downregulation of RIP140 by a siRNA in the double-stimulation condition.
- **E2:** Estradiol stimulation
- **RA:** Retinoic Acid stimulation

**Basal line:**

- **E2+RA** The double stimulation condition

**Perturbation rules:**

- **E2+RA+siLCoR->LCoR**
- **E2+RA+siRIP140->siRIP140**
- **E2->Hoxa5** (Switch from E2+RA to E2)
- **RA->Luciferase** (Switch from E2+RA to RA)
- **E2+RA->0** (The basal line)

Rules can be defined as a character vector and then transformed into a perturbation matrix as follows:

```
rules=c("E2+RA+siLCoR->LCoR","E2+RA+siRIP140->siRIP140","E2->Hoxa5","RA->Luciferase","E2+RA->0")
matp=read.rules(rules)
```

or they can be read from a .csv file.

```
#Example of reading perturbation rules from a specific folder
rules=read.rules(/User/some_path_to_folder/rules_file.csv)
```

The output of the function will be the peturbation matrix:

```
#>            E2+RA+siLCoR E2+RA+siRIP140 E2 RA E2+RA
#> LCoR                  1              0  0  0     0
#> siRIP140              0              1  0  0     0
#> Hoxa5                 0              0  1  0     0
#> Luciferase            0              0  0  1     0
```

NOTE: Modules and perturbations must have the same names as the row and column names in experimental data tables (names are case sensitive).

### Calculation of the connectivity map

Once the data tables are in the right format and the perturbation matrix has been created the next step is to calculate the connectivity map of the network. For this example we chose to average the six data tables using the `data2sdmean()` function.

```
data=data2sdmean(data)
```

This function creates a list containing the standard deviation and average values of a data set. Namely, if n replicates were generated, then n data tables should be provided to `data.setup` and `data2sdmean` computes the average and standard deviation of each cell over the n replicates.

Finally we compute the connectivity map using the `mra()` function.

```
res=mra(data$mean,matp)
```

### Plot of the connectivity map

The `netgraph()` function plots a graphical version of the network connectivity map returned by the `mra()` function.

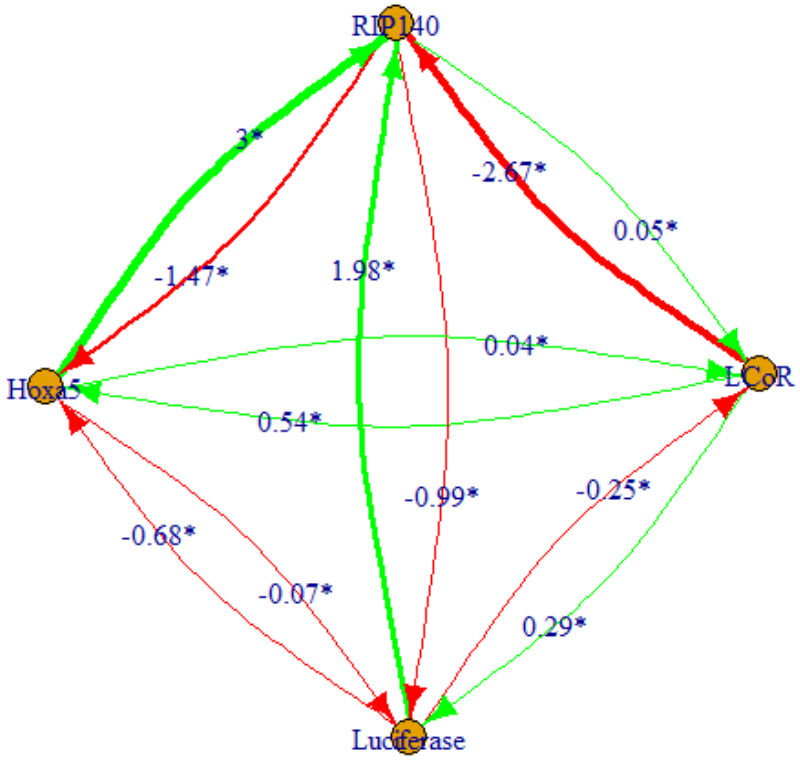
```
netgraph(map)
```

Estimation of confidence intervals for each connectivity coefficient is obtained using the `interval()` function.

```
inter=interval(data$mean,data$sd,matp,nrep=6)
```

The output of this function allows to add an asterik on each coefficient for which zero is not included within the confidence interval.

```
netgraph(map,inter=inter)
```




The netgraph function have many arguments parameters that can be changed by the user to fit his needs (see function **netgraph()** details.)

# Functions

- **data.setup()**
- **data2sdmean()**
- **interval()**
- **mra()**
- **netgraph()**
- **read.rules()**

---

## data.setup()

Data setup for Modular Response Analysis

### Description

Checks for possible errors in tables of experimental data and in the perturbation matrix. Returns data properly formated for the mra function.

### Usage

```
data.setup(obj, pert.tab = NULL, dec = ".")
```

### Arguments

| Argument | Description |
|---|---|
| `obj` | A character string specifying the path to a folder that contains the experimental data tables and the perturbation rules, or a list containing several data tables, or a single data table. |
| `pert.tab` | Optional. The perturbation rule table if not provided in `obj`. |
| `dec` | The decimal separator as in `base::read.csv` (default is ".") |

### Value

If more than one data table are checked, then a list of matrices in the correct format for the mra function is returned. If a single table is checked, then a single matrix in the right format is returned. It also checks for correspondance with the perturbation rules table.

### Examples

```
#Example using q-PCR data stored in the package files
data=data.setup(list(estr1_A,estr1_B,estr2_A,estr2_B,estr3_A,estr3_B))
#Examples using a gene network from Kholodenko et.al. PNAS 2002 figure a and b
#Data was obtained from the given R matrix
```

```
data.A=data.setup(gene.network.A)
data.B=data.setup(gene.network.B)
```

---

## data2sdmean()

Standard deviation and average calculation of a data set.

### Description

This function computes the standard deviation and average values of a data set. Namely, if n replicates were generated, then n data tables should be provided to `data.setup` and `data2sdmean` computes the average and standard deviation of each cell over the n replicates.

### Usage

```
data2sdmean(x)
```

### Arguments

| Argument | Description |
|---|---|
| x | A list containing 2 or more data tables. |

### Value

A list containing two matrices, the standard deviations and mean values for each variable in x.

### Examples

```
data=data.setup(list(estr1_A,estr1_B,estr2_A,estr2_B,estr3_A,estr3_B))
data2sdmean(data)
```

---

## interval()

Confidence interval calculation for MRA connectivity coefficients

### Description

A function that estimates confidence intervals of MRA connectivity coefficients based on replicates. It uses a bootstrap algorithm.

### Usage

```
interval(tab,sd.tab,matp,n=10000,nrep=2)
```

### Arguments

| Argument | Description |
|---|---|
| tab | A data table containing the experimental data in the format output by data.setup. |
| sd.tab | A data table containing the standard deviation values. |
| matp | A perturbation rule table, with rows corresponding to the MRA modules and columns to perturbations. |
| n | Number of samples for the bootstrap algorithm. |
| nrep | Number of replicates. Default is 2. |

### Value

A list containing the upper and lower values of the confidence interval of each connectivity coefficient.

### Examples

```
data=data.setup(list(estr1_A,estr1_B,estr2_A,estr2_B,estr3_A,estr3_B))

# we first average the technical replicates of each biological replicate
# and then only keep the (averaged) biological replicates for the calculation
CODE
sd.mean=data2sdmean(data)
rules=c("Et->Luciferase","E2+siRIP140->RIP140","E2+siLCoR->LCoR","E2->0")
matp=read.rules(rules)
interval(sd.mean$mean,sd.mean$sd,matp,nrep=6)
```

---

## mra()

Modular Reponse Analysis function

### Description

Calculation of connectivity coefficients between MRA modules.

### Usage

```
mra(tab,matp,check=TRUE)
```

### Arguments

| Argument | Description |
|---|---|
| tab | Data in the format returned by data.setup or a single data table in the format required by data.setup. |
| matp | A perturbation rule table, with rows corresponding to the MRA modules and columns to perturbations. |
| check | Logical. If TRUE then data and perturbation rule are checked for consistency (by calling data.setup). |

### Value

A list containing the connectivity map, the local responses matrix, the network responses matrix to perturbations and the basal line for all the modules.

### Examples

```
#It creates the connectivity map between 2 transcriptional nuclear coregulators
#(RIP140 and LCoR) and estrogen receptor alpha transcriptional activity reported by
#a luciferase gene. q-PCR data is stored in the package files used below in the function data.setup
#The model is obtained using the E2 stimulated condition.

data=data.setup(list(estr1_A,estr1_B,estr2_A,estr2_B,estr3_A,estr3_B))
sd.mean=data2sdmean(data)
rules=c("Et->Luciferase","E2+siRIP140->RIP140","E2+siLCoR->LCoR","E2->0")
matp=read.rules(rules)
mra(sd.mean$mean,matp,check=TRUE)
```

---

## netgraph()

Plot of network connectivity map

### Description

Plots a graphic representation of a biological network connectivity map calculated by MRA.

### Usage

```
netgraph(map,layout=igraph::layout_with_kk,pertu=FALSE,inter=NULL,
      cutoff=NULL,main=NULL,digits=2,v.type="circle",p.type="rectangle",
      outfile=NULL, ...)
```

### Arguments

| Argument | Description |
|---|---|
| `map` | A list containing the connectivity map (link matrix) of a network, and local matrix responses to perturbations, i.e., the output of `mra`. |
| `layout` | Either a data.frame containing the x and y coordinates and the color of the vertices in the network, or a layout function according to the igraph package. Default uses the "layout_with_kk" igraph layout. |
| `pertu` | A boolean specifying whether perturbations should be plotted as vertices in the network. |
| `inter` | Confidence intervals calculated by `interval`; connectivity coefficient with a confidence interval that does not include 0 are deemed significant and marked by an asterisk. |
| `cutoff` | Minimum value for a connectivity link coefficient between two modules for being plotted. |
| `main` | A character string giving the title of the plot. |

| Argument | Description |
|---|---|
| digits | Number of digits to be plotted for each connectivity coefficient. |
| v.type | Vertex type for biological modules. One of “none”, “circle”, “square”, “csquare”, “rectangle” “crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”. |
| p.type | Vertex type for perturbations. One of “none”, “circle”, “square”, “csquare”, “rectangle” “crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”. |
| outfile | Optional. A character string giving the name of the output file in graphML format. |
| ... | Arguments to be passed to igraph plot such as vertex and edges plotting parameters. |

### Value

If layout is an igraph function, then a data.frame containing the coordinates of the layout selected is returned. If outfile is not NULL, then the network is written in the graphML file format.

### Examples

```
data=data.setup(list(estr1_A,estr1_B,estr2_A,estr2_B,estr3_A,estr3_B))
sd.mean=data2sdmean(data)
rules=c("Et->Luciferase","E2+siRIP140->RIP140","E2+siLCoR->LCoR","E2->0")
matp=read.rules(rules)
map=mra(sd.mean$mean,matp,check=FALSE)
inter=interval(sd.mean$mean,sd.mean$sd,matp,nrep=6)
netgraph(map,pertu=TRUE,inter=inter)
```

---

## read.rules()

Creates a perturbation matrix from a set of perturbation rules.

### Description

A function that takes a set of perturbation rules as input and creates a perturbation matrix to be used in other functions of the package.

### Usage

```
read.rules(obj)
```

### Arguments

| Argument | Description |
|---|---|
| obj | Either the path to a perturbation rules file without header, or a vector of strings specifying the perturbation rules (see details). |

### Details

Perturbation rules are a set of strings specifying the action of perturbations upon modules in a network. In both cases, modules and perturbations must have the same names as the row and column names in experimental data tables (names are case sensitive).

The rules syntax is `Perturbation->Module` for a single perturbation,`Perturbation->0` to specify the basal line and `0->Module` to specify modules that were not perturbated, but for which connectivity issued from other modules exist (unidirectional MRA). At least two rules and a basal line must be defined.

### Examples

```
rules=c("Et->Luciferase","E2+siRIP140->RIP140","E2+siLCoR->LCoR","E2->0")
```