Software for motif analysis (ERDF project 1.1.1.1/16/A/135, WP3)

Short description

Purpose

Software modules are designed for collecting statistics about appearance of various motifs (general or composite) in the gene and protein interaction data and preparing data for visualization of a various kind. Results obtained using this software are described in the paper "Graph-based network analysis of transcriptional regulation pattern divergence in duplicated yeast gene pairs". This software serves as a supplementary material for the paper and allows to reconstruct completed experiments using the same data source or provide similar experiments using other data sets. Software package contains randomization and statistics modules together with the corresponding data files.

Requirements

Python version 3.6 or above is required.

Modules

1. Randomization module

Module prepares data set for the upcoming analysis and counts number of appearances of the particular motif. Processing time may be quite large (one iteration on Intel Core i5 processor takes 8 seconds), therefore is implemented option to disrupt calculations and save intermediate results after predefined number of iterations.

Contents

Python scripts in `Utils` folder (`utilsRandomize.py` and `utilReader.py`) and `runRandomize.py`, data files.

Usage

In the text of the script `runRandomize.py` values for several parameters should be specified:

- *fileInput* - path to the JSON file containing description of the links.
- *fileInputProteins* - path to the JSON file containing list of the proteins to be analyzed.
- *fileOutput* - path to the file where result will be stored as JSON file.
- *randomizeCount* - total number of iterations.
- *saveAfterCount* - number of iterations after what intermediate results will be stored.

Example of `runRandomize.py`:

```
import Utils.utilsRandomize

templ = { \
        "fileInput": "./Data/links.json", \
        "fileInputProteins": "./Data/proteinList.json", \
        "fileOutput": "./Results/dataRandomized.json", \
        "randomizeCount": 12, \
        "saveAfterCount": 3, \
}
```

```
rm = Utils.utilsRandomize.randomizeMotifs(templ)
rm.executeRandomize()
```

Description of links contains list of pairs “protein - gene”.

For example: `[["abf1", "TRM5"], ["rsc2", "YKR104W"], ["stp2", "TIR1"],...]` Proteins and genes are coded in a standard way and can contain only lowercase letters and digits, while gene names - only uppercase letters and digits.

List of proteins to be analyzed contains list of protein names in the described format, i.e. `["sir2", "hst1", "rsc2", "rsc1", "stb6",...]`

After executing “`Python runRandomize.py`”, the specified data file is created. If execution is interrupted, then in the file will be intermediate results and, in the case of script re-run, this file will be used as a source for the further calculations.

With the example parameters if there is already intermediate result file after two iterations, output will be the following (there are performed 14 iterations instead of 12):

```
DONE load  ./Results/dataRandomized.json
DONE load  ./Data/links.json
DONE load  ./Data/proteinList.json
Continue randomize from 2
DONE save  ./Results/dataRandomized.json
After save rand count 5
DONE save  ./Results/dataRandomized.json
After save rand count 8
DONE save  ./Results/dataRandomized.json
After save rand count 11
DONE save  ./Results/dataRandomized.json
After save rand count 14
DONE save  ./Results/dataRandomized.json
Randomization is completed 14
```

2. Statistics module

Module produces comma-separated data file of motif statistics. Motif is labeled as *relevant* if in randomized graphs it appears fewer times than specified by a threshold value.

Contents

Python scripts in `Utils` folder (`utilsStatistics.py` and `utilsReader.py`) and `runStatistics.py`, data files.

Usage

In the text of the script `runStatistics.py` values for several parameters should be specified:

- *fileInputLinks* - path to the JSON file containing description of the links.

- *fileMotifRandomized* - path to the JSON file containing result of the randomization.
- *fileInputOhnologue* - path to the JSON file containing description of ohnologues and paralogues.
- *fileOutputStats* - path to the file where result will be stored as comma-separated file.
- *ratio* - threshold value - number of motif appearances to the total number of iterations.

Example of `runStatistics.py`:

```
import Utils.utilsStatistics

templStats = { \
    "fileInputLinks": "./Data/links.json", \
    "fileMotifRandomized": "./Results/dataRandomized.json", \
    "fileInputOhnoloque": "./Data/ohnologues.json", \
    "fileOutputStats": "./Results/statistics.csv", \
    "ratio": 0.0005 \
}

ms = Utils.utilsStatistics.utilsMotifs(templStats)
ms.finalStats()
```

After executing "`Python runStatistics.py`" the specified comma-separated file is created.

Result is in the following format:

protein-1	protein-2	value	type	links	FFL-X-1	FFL-X-2	FFL-X-12	FFL-X-1-R	FFL-X-2-R	FFL-X-12-R	FFL-Y-1	FF
sir2	hst1	545	ohnologue	1723	13989	171	0	7582	156	0	5392	18
rsc2	rsc1	434	ohnologue	1429	2240	2162	63	1380	1396	24	6271	42
stb6	stb2	401	ohnologue	271	14	290	0	8	200	0	1073	18
pip2	oaf1	374	ohnologue	2192	9604	6948	938	5182	4561	191	8486	48

Each row contains two protein identifiers, protein similarity value, type (ohnologue or paralogue), number of links containing protein-1 or protein-2, number of particular motifs containing particular protein vertex (vertices). If column name contains "R", corresponding value denotes relevant values in the corresponding motif (column name without "-R").

Motif names are coded using the format `<Gen_name>-<position>-<influence>`, where:

<Gen_name> is generalized motif name (`FFL`, `DFFLUp`, `DFFLDown`, `C2`, `C3` or `C4`);
<position> is selection position (`X`, `Y` or `Z`);
<influence> denotes protein influence in the motif (`1` - protein-1 only, `2` - protein-2 only, `12` - both).