Software for predicting missing protein values from RNA values

Short description

Software consists of two Python modules:

1. "DLNetworkForProteinAbundancePreparation.py" and
2. "DLNetworkForProteinAbundancePrediction.py".

The first one is designed for preparation of gene and protein data from a raw data to establish one-to-one correspondence between identifiers and tissue names in these two files. In general it should be used to synchronize two csv-style files with headers. The prediction module uses prepared data for predicting missing protein values from the RNA values and optional gene context file.

Software is tested with Python version 3.7.8 with package TensorFlow version 1.15.

Important! *Software will not work with TensorFlow 2!*

Technical details are described below.

1. Module "DLNetworkForProteinAbundancePreparation.py"

Input

File with gene information and file with protein information. Each file should be "csv or tsv style" text file with header row, titles and values should be delimited by some delimiter. Delimiters for both files may be different. In both files name of the column containing identifiers should be the same. If output files should be with the same delimiter different from original files, it should be specified.

Output

Two files – file corresponding to gene file and file corresponding to protein file. Only columns appearing in both input files will be present in output files. Order of columns will be the same in both files. First column will be the identifier column. Other columns are in the alphabetical order. Only rows having identifiers present in both input files, will be written in output files. Rows are sorted in ascending order by identifier column values.

Parameters

Name	Description	Mandatory
`-i, --i1`	The name of the first input file	YES
`-I, --i2`	The name of the second input file	YES
`-o, --o1`	The name of the first output file	YES
`-O, --o2`	The name of the second output file	YES
`-d, --d1`	Delimiter in the first input file (default – TAB)	NO
`-D, --d2`	Delimiter in the second input file (default – TAB)	NO
`-e, --id`	Name of the identifier column	YES
`-f, --dout`	Delimiter for the output files	NO
`-c, --cs`	Case sensitive	NO
`-S, --subst`	Substitute missing values with zeroes (default - NO)	NO
`-n, --nthr`	Value of the normalisation threshold (if stated, data will be normalised)	NO

Parameters are specified by the corresponding keyword, order of parameters may be arbitrary. Delimiters should be one character (tabulation character is denoted as "TAB").

Normalisation

If normalisation threshold is stated (it can be also 0.0, in fact not influencing output for the first data set) normalisation is done according to the fallowing algorithm:

1. After common columns and corresponding rows in two datasets are determined, there are found columns where in both data sets are just numerical values (empty cells are assumed to contain zeroes).
2. All values less than threshold value in the first data set is substituted by zeroes.
3. Average of all non-zero values in the first data set avg_1 is calculated.
4. Average of all non-zero values in the second data set avg_2 is calculated.
5. All numerical values in the second data set are multiplied by avg_1/avg_2.

Substitution of missing values

If just substitution of missing values with zeroes is necessary, parameter "`--subst`" or "`-S`" should be specified.

Usage

```
python DLNetworkForProteinAbundancePreparation.py <list of
parameters>
```

Usage examples

Example 1

```
Python DLNetworkForProteinAbundancePreparation.py –i a.csv –d’,’
–o a1.csv --i2 b.tsv --o2 b1.tsv --id "Gene.ID"
```

Comma-separated file `a.csv` and tab-delimited file `b.tsv` are processed using values from the column "`Gene.ID`" as identifiers. Result is written into files `a1.csv` and `b1.tsv` with the delimiters comma and TAB, correspondingly. Processing is case insensitive.

Example 2

```
Python DLNetworkForProteinAbundancePreparation.py --d1 ";" --i1
"data1.txt" --i2 "data2.txt" --o1 "d1.out" --o2 "d2.out" --d2 ","
--id "Gene stable ID" --dout "TAB"
```

Semicolon-separated file `data1.txt` and comma-separated file `data2.txt` are processed using values from the column "`Gene stable ID`" as identifiers. Result is written into tab-separated files `d1.out` and d2.out, correspondingly. Processing is case insensitive.

Example 3

```
Python DLNetworkForProteinAbundancePreparation.py -i CR_Gena.csv
-I CR_proteome.csv -o Gena_out.csv -O Proteome_out.csv -e
"Gene.ID"
```

Tab-separated files `CR_Gena.csv` and `CR_proteome.csv` are processed using values from the column "`Gene.ID`" as identifiers. Result is written into tab-separated files `Gena_out.csv` and `Proteome_out.csv`, correspondingly. Processing is case insensitive.

Example 3a

```
Python DLNetworkForProteinAbundancePreparation.py -i CR_Gena.csv
-I CR_proteome.csv -o Gena_out.csv -O Proteome_out.csv -e
"Gene.ID" -n 0.5
```

Example analogous to the previous with the additional normalisation of the output (thresholding of values in the first data set and multiplying values in the second).

Example 4

```
Python DLNetworkForProteinAbundancePreparation.py -i AAA.txt -I
BBB.txt -o AAA_out_cs.csv -O BBB_out_cs.csv -e "Gene" -c 1>
diffqn_cs 2> differrqn_cs
```

Tab-separated files `AAA.txt` and `BBB.txt` are processed using values from the column "`Gene`" as identifiers. Result is written into tab-separated files `AAA_out_cs.csv` and `BBB_out_cs.csv`, correspondingly. Processing is case insensitive. Output messages are written into the file `diffqn_cs` while errors (if any) into `differrqn_cs`.

Contents of the `diffqn_cs` are the following:

```
[('-i', 'AAA.txt'), ('-I', 'BBB.txt'), ('-o', 'AAA_out_cs.csv'),
('-O', 'BBB_out_cs.csv'), ('-e', 'Gene'), ('-c', '')]
Input file 1 is "AAA.txt"
Input file 2 is "BBB.txt"
Output file 1 is "AAA_out_cs.csv"
Output file 2 is "BBB_out_cs.csv"
Delimiter for file 1 is "   "
Delimiter for file 2 is "   "
Identifier is "Gene"
Case sensitive is "True"
Column "Igrov1" is not present in the file "BBB.txt" header.
Column "x" is not present in the file "AAA.txt" header.
Column "CCRFCEM" is not present in the file "AAA.txt" header.
Column "NCIH23" is not present in the file "AAA.txt" header.
Column "HCC2998" is not present in the file "AAA.txt" header.
Column "IGROV1" is not present in the file "AAA.txt" header.
Column "M14" is not present in the file "AAA.txt" header.
Column "MALME3M" is not present in the file "AAA.txt" header.
Column "MCF71" is not present in the file "AAA.txt" header.
Column "MDAMB231" is not present in the file "AAA.txt" header.
Column "MOLT4" is not present in the file "AAA.txt" header.
Column "OVCAR3" is not present in the file "AAA.txt" header.
Column "RXF393" is not present in the file "AAA.txt" header.
Column "SF295" is not present in the file "AAA.txt" header.
Column "SN12C" is not present in the file "AAA.txt" header.
Column "SNB19" is not present in the file "AAA.txt" header.
Column "SNB75" is not present in the file "AAA.txt" header.
Column "U251MG" is not present in the file "AAA.txt" header.
Column "UACC62" is not present in the file "AAA.txt" header.
Column "UO31" is not present in the file "AAA.txt" header.
Collected and sorted column names: ['Gene', 'A498', 'A549',
'ACHN', 'BT549', 'COLO205', 'Caki1', 'DU145', 'EKVX', 'HCT116',
'HCT15', 'HL60', 'HOP62', 'HOP92', 'HT29', 'Hs578T', 'K562',
'KM12', 'LOXIMVI', 'MCF7', 'MDAMB435', 'NCIH226', 'NCIH322T',
'NCIH460', 'NCIH522', 'OVCAR4', 'OVCAR5', 'OVCAR8', 'PC3',
'RPMI8226', 'SF268', 'SF539', 'SKMEL2', 'SKMEL28', 'SKMEL5',
'SKOV3', 'SR', 'SW620', 'T47D', 'TK10', 'UACC257', 'X786O']
Identifier "eNSG00000001084" appears more than once in "AAA.txt"
or is not present in "BBB.txt".
Identifier "ENSG00000030066" appears more than once in "AAA.txt"
or is not present in "BBB.txt".
Identifier "ENSG00000033050" appears more than once in "AAA.txt"
or is not present in "BBB.txt".
Identifier "ENSG00000071894" appears more than once in "AAA.txt"
or is not present in "BBB.txt".
Identifier "ENSG00000085063" appears more than once in "AAA.txt"
or is not present in "BBB.txt".
... more warnings...
```

Highlighted are values which have corresponding values written in uppercase. If the script would be executed in a case insensitive mode, these will not be reported as errors.

Error messages and warnings

Errors

`Identifier <identifier_name> is not present in the file <input_file_name> header.`

The identifier specified in parameters is not specified in the input file header.

`Data row length exceeds length of the header row in <input_file_name>.`

Number of fields in header row is less than in data row.

Ordinary system errors (like “File not found”) are also possible.

Warnings

`Column <column_name> is not present in the file <input_file_name> header.`

The column name specified in one of input files is not present in the other - the mentioned column will not be present in the output file produced from this input file.

`Identifier <identifier_value> is not present in <input_file_name>.`

Identifier field value specified in one of input files is not present in the other - row with the mentioned identifier value will not be present in the output file produced from this input file. Only five warnings of this kind will be reported. In case of more errors, warnings are tailed with “`... more warnings ...`”.

2. Module "DLNetworkForProteinAbundancePrediction.py"

Input

File with RNA information and file with protein information. Each file should be tab-delimited text file with header row. In both files name of the column containing identifiers should be the same. Optional gene context file may be specified. If context file is not specified, obtained results are close to obtained by linear regression. In the file with protein information missing values should be specified as 0 or NA.

Output

Output file is corresponding to the protein file where missing values are replaced by calculated ones.

Usage

```
python DLNetworkForProteinAbundancePrediction.py <RNA_file>
<protein_file> <out_file> [<context_file>]
```

3. Complex software usage example

We will demonstrate usage of software using the following TAB-separated data files (paths to the files are relative from the file containing Python programs and demo .bat and .sh scripts):

`RNA file: data_sets_for_testing/E_MTAB_2836_fpkms.tsv`

`Protein file: data_sets_for_testing/E_PROT_29.tsv`

`Gene annotations: data/gene_annotations_human.txt`

The files `E_MTAB_2836_fpkms.tsv` and `E_PROT_29.tsv` contain correspondingly RNA and protein expression data as exported from Expression Atlas, however, header lines with comments should be removed (a single header line with column names should be present) and column names in one or both files should be manually curated to provide consistent and unique mapping between tissues or cell lines for which both protein and RNA expression data is available (the tissues or cell lines for which no mapping will be found will be removed by the program in this pre-processing stage). (E.g., if one of the Expression Atlas export files contains a column "`animal ovary`" and another file a column "`ovary`", which is deemed to refer to the same tissue, a consistent naming for these columns should be provided manually.)

Data will be prepared using command:

```
python DLNetworkForProteinAbundancePreparation.py --i1
data_sets_for_testing/E_MTAB_2836_fpkms.tsv --o1
data_sets_for_testing/tissue29_rna_test_sample_2.tsv --i2
data_sets_for_testing/E_PROT_29.tsv --o2
data_sets_for_testing/tissue29_prot_test_sample_2.tsv --id "GeneID" --cs
--nthr "0.5" 1>data_sets_for_testing/normalization_messages_sample_2.txt
2>data_sets_for_testing/normalization_error_messages_sample_2.txt
```

Data in RNA and protein files will be synchronized removing information about genes and/or features residing just in one data file. Correspondence should be absolute. For example, "Gene

Name” in one file and “Gene_Name” in another are considered as different and, as a result, this column is not included in the resulting files.

File “normalization_messages_sample_2.txt” contains diagnostic information:

```
[('--i1', 'data_sets_for_testing/E_MTAB_2836_fpkms.tsv'), ('--o1',
'data_sets_for_testing/tissue29_rna_test_sample_2.tsv'), ('--i2',
'data_sets_for_testing/E_PROT_29.tsv'), ('--o2',
'data_sets_for_testing/tissue29_prot_test_sample_2.tsv'), ('--id',
'GeneID'), ('--cs', ''), ('--nthr', '0.5')]
=> Input file 1 is "data_sets_for_testing/E_MTAB_2836_fpkms.tsv"
=> Output file 1 is
"data_sets_for_testing/tissue29_rna_test_sample_2.tsv"
=> Input file 2 is "data_sets_for_testing/E_PROT_29.tsv"
=> Output file 2 is
"data_sets_for_testing/tissue29_prot_test_sample_2.tsv"
=> Identifier is "GeneID"
=> Case sensitive is "True"
=> Normalization threshold is "0.5"
Input file 1 is "data_sets_for_testing/E_MTAB_2836_fpkms.tsv"
Input file 2 is "data_sets_for_testing/E_PROT_29.tsv"
Output file 1 is "data_sets_for_testing/tissue29_rna_test_sample_2.tsv"
Output file 2 is "data_sets_for_testing/tissue29_prot_test_sample_2.tsv"
Delimiter for file 1 is "     "
Delimiter for file 2 is "     "
Identifier is "GeneID"
Case sensitive is "True"
Substitute missing values with zeroes is "False"
Normalize is "True" (threshold is "0.5")
Column "Gene_Name" is not present in the file
"data_sets_for_testing/E_PROT_29.tsv" header.
Column "cerebral.cortrex" is not present in the file
"data_sets_for_testing/E_PROT_29.tsv" header.
Column "skin" is not present in the file
"data_sets_for_testing/E_PROT_29.tsv" header.
Column "skeletal.muscle" is not present in the file
"data_sets_for_testing/E_PROT_29.tsv" header.
Column "Gene Name" is not present in the file
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" header.
Column "1" is not present in the file
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" header.
Column "brain" is not present in the file
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" header.
Column "pituitary.hypophysis" is not present in the file
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" header.
Collected and sorted column names: ['GeneID', 'adipose.tissue',
'adrenal.gland', 'appendix', 'bone.marrow', 'colon', 'duodenum',
'endometrium', 'esophagus', 'fallopian.tube', 'gall.bladder', 'heart',
'kidney', 'liver', 'lung', 'lymph.node', 'ovary', 'pancreas',
'placenta', 'prostate', 'rectum', 'salivary.gland', 'small.intestine',
'smooth.muscle', 'spleen', 'stomach', 'testis', 'thyroid', 'tonsil',
'urinary.bladder']
Potentially numerical columns: ['adipose.tissue', 'adrenal.gland',
'appendix', 'bone.marrow', 'colon', 'duodenum', 'endometrium',
'esophagus', 'fallopian.tube', 'gall.bladder', 'heart', 'kidney',
'liver', 'lung', 'lymph.node', 'ovary', 'pancreas', 'placenta',
'prostate', 'rectum', 'salivary.gland', 'small.intestine',
'smooth.muscle', 'spleen', 'stomach', 'testis', 'thyroid', 'tonsil',
'urinary.bladder']
Final list of numerical columns: ['adipose.tissue', 'adrenal.gland',
'appendix', 'bone.marrow', 'colon', 'duodenum', 'endometrium',
'esophagus', 'fallopian.tube', 'gall.bladder', 'heart', 'kidney',
'liver', 'lung', 'lymph.node', 'ovary', 'pancreas', 'placenta',
'prostate', 'rectum', 'salivary.gland', 'small.intestine',
'smooth.muscle', 'spleen', 'stomach', 'testis', 'thyroid', 'tonsil',
'urinary.bladder']
Identifier "ENSG00000000005" appears more than once in
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" or is not present in
"data_sets_for_testing/E_PROT_29.tsv".
Identifier "ENSG00000001631" appears more than once in
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" or is not present in
"data_sets_for_testing/E_PROT_29.tsv".
Identifier "ENSG00000002016" appears more than once in
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" or is not present in
"data_sets_for_testing/E_PROT_29.tsv".
Identifier "ENSG00000002079" appears more than once in
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" or is not present in
"data_sets_for_testing/E_PROT_29.tsv".
Identifier "ENSG00000002745" appears more than once in
"data_sets_for_testing/E_MTAB_2836_fpkms.tsv" or is not present in
"data_sets_for_testing/E_PROT_29.tsv".
... more warnings...
Non_zero_sum1 = 6736631.79999997   non_zero_count1 = 302570
Non_zero_sum2 = 23555349601.52263   non_zero_count2 = 288104
Normalisation coefficient = 0.0002723182073538691
```

In successful run file “`normalization_error_messages_sample_2.txt`” should be empty.

Obtained files can be further used for prediction:

```
python DLNetworkForProteinAbundancePrediction.py
data_sets_for_testing/tissue29_prot_test_sample.tsv
data_sets_for_testing/tissue29_rna_test_sample.tsv
data_sets_for_testing/prediction_results_sample_2.txt
data/gene_annotations_human.txt
1>data_sets_for_testing/prediction_messages_sample_2.txt
2>data_sets_for_testing/prediction_error_messages_sample_2.txt
```

Since “tensorflow” version is newer than version used during writing the module, there will be quite a long list of warnings about outdated constructions in “`prediction_error_messages_sample_2.txt`”.

File `prediction_messages_sample_2.txt`” will contain diagnostic output:

```
onto items: [447, 427, 2858, 252]
dimensions =  3984
tissue types =  29
train set length: 302570
Computing...
step =  5000 loss =  0.8645789566338062 time= 16.56722927093506
step =  10000 loss =  0.8044025776825845 time= 16.688429832458496
step =  15000 loss =  0.7053230633050204 time= 16.816829442977905
step =  20000 loss =  0.6879547361850739 time= 16.958229780197144
step =  25000 loss =  0.6764348643571139 time= 17.75283122062683
step =  30000 loss =  0.6621739984840155 time= 17.624431610107422
step =  35000 loss =  0.6619453135758638 time= 17.083029985427856
step =  40000 loss =  0.6539775541782379 time= 17.04723048210144
step =  45000 loss =  0.6543481953769922 time= 16.593028783798218
step =  50000 loss =  0.6503962195456028 time= 17.670031309127808
step =  55000 loss =  0.6503952523231507 time= 16.785629749298096
step =  60000 loss =  0.6457665252953768 time= 18.44151782989502
Predicting...
Done
```

The result is written in the file “`prediction_results_sample_2`”.