⚠ This is a Work in Progress document. Be warned that it might contain mistakes, inaccuracies and have missing chapters.

Contents

1 Language **1**
1.1 Language description . . . 1
1.2 Modules . . . 1
1.3 Comments . . . 2
1.4 Begin/End . . . 2
1.5 Variables . . . 3
1.6 Loops . . . 4
1.6.1 n-loop (repeat block n times) . . . 4
1.6.2 Iteration with variable . . . 4
1.6.3 Loop over a set of values . . . 5
1.7 Module description file . . . 5

2 Modules **7**
2.1 Multivariate analysis . . . 7
2.1.1 PCA (Principal Component Analysis) . . . 7
2.1.2 PCO (Principal Coordinates Analysis) . . . 8
2.1.3 SVD (Singular Value Decomposition) . . . 9
2.2 Metrics . . . 10
2.2.1 Euclidean metric . . . 10
2.2.2 Minkowski metric . . . 11
2.2.3 Jaccard index . . . 12
2.2.4 Jaccard-Steinhaus index . . . 13
2.2.5 Hamming distance . . . 14
2.2.6 Manhattan distance . . . 15
2.2.7 p-distance . . . 16
2.2.8 Jukes-Cantor distance . . . 17
2.2.9 Kimura two-parameter distance . . . 18
2.2.10 Spectrum distance . . . 19
2.2.11 Mantel test . . . 21
2.2.12 Rank Mantel test . . . 22
2.2.13 Correlation . . . 23
2.2.14 Compression metric . . . 24
2.3 Matrix operations . . . 26
2.3.1 Centering . . . 26
2.3.2 Double centring . . . 27
2.3.3 Normalize length . . . 28
2.3.4 Matrix on elements operations . . . 29
2.3.5 Compute statistics . . . 31

2.4 Data preparation . . . 32
2.4.1 Transpose . . . 32
2.4.2 Table row selection . . . 33
2.4.3 Table column selection . . . 36
2.4.4 Append matrix to the right . . . 37
2.4.5 Append matrix to the bottom . . . 38
2.4.6 Append file . . . 39
2.4.7 Make binary matrix . . . 40
2.4.8 Matrix shifting . . . 41
2.4.9 bootstrap . . . 42
2.4.10 bootstrap labels . . . 43
2.4.11 Row permutation . . . 44
2.5 File operations . . . 45
2.5.1 Copy file . . . 45
2.5.2 Delete file . . . 46

3 Methods **47**
3.1 Principal Coordinates . . . 47

Index **a**

Chapter 1

Language

1.1 Language description

It is assumed the user would use Microsoft Excel or any other application they are familiar with to compose a script. It should be saved as a CSV file with semicolon delimeter. The format is widespred making it easy to compose script in almost any editor.

The script consists of calls to computation modules and control structures possibly sprinkled with comments.

1.2 Modules

Modules in JACOBI 4 scripts are data processing units. Each module requires a set of parameters: input files, output files and maybe some additional data.

Each script is validated prior to its execution to diagnose possible syntax errors or issues related to module invocations. For example, if module requires three arguments but only two were provided, the script wouldn't be launched.

A guideline for using modules can be summarized as follows:

- module name and all of its parameters should be placed in separate cells;
- module name should be followed by its parameters;
- parameters should be placed on the same line as the module name;
- parameters should be placed in accordance with the documentation;
- the script should be saved in a CSV file.

Module use example:

transpose	matrixA.csv	out.csv

It might be interesting to know for the advanced user that modules are executable files located in the "bin" folder. It is possible to use them from command line.

The main idea of composing a script is to have a series of module invocations with the output of one module being used by the following one.

1.3 Comments

Comments start with "//". All symbols till the end of the line are considered to be a comment and ignored.

An example of a line with comment:

bootstrap	file1.csv	file2.csv//bootstrap module	invocation

Substring "//bootstrap module" and the following cells are ignored.

1.4 Begin/End

JACOBI 4 was created for stream data procesing. In practice this means the user can write single script for processing a set of files. The script is written and debugged once for multiple uses afterwards.

Keywords "begin" and "end" are used to define a section of the script for execution. The keywords should be used in the first row and should be the only ones on the line. Everything outside of it is ignored and does not participate in validation.

This is useful for debugging purposes to limit execution scope to verify its correctness.

Lets consider the following simple example [1]. There are multiple files with raw experimental data that requires preprocessing for further analysis. We'll assume they are saved as CSV files and we have to remove lines with cells containing non-numeric values, transpose and normalize matrix and calculate logarithm for each value.

First off, we have to make sure the script works correctly for one file. The preprocessing script for input file raw.csv could be written as follows:

deleteAllRowsWithNotNumberValues	raw.csv	nums.csv	non.csv
transpose	nums.csv	transp.csv	
normalizeLength	transp.csv	norm.csv	
log	norm.csv	log.csv	2

Issues related to input data may surface during script execution. After the issue is fixed it is possible to execute only pertinent part of the script using "begin-end" block as follows:

deleteAllRowsWithNotNumberValues	raw.csv	nums.csv	non.csv
transpose	nums.csv	transp.csv	
BEGIN			
normalizeLength	transp.csv	norm.csv	
log	norm.csv	log.csv	2
END			

During script execution everything before "BEGIN" and after "END" will be ignored.

There could be multiple "BEGIN-END" sections in one script but they cannot be nested.

[1] Examples in this section are contrived. They are ment to illustrate the use of JACOBI 4 scripting language.

Correct block usage

mobule 1
mobule 2
BEGIN
mobule 3
END
mobule 4
mobule 5
mobule 6
BEGIN
mobule 7
mobule 8
END
mobule 9

Incorrect block usage

mobule 1
END
mobule 2
BEGIN
mobule 3
BEGIN
mobule 4
BEGIN
END
mobule 5
mobule 6
BEGIN
mobule 7
END
mobule 8
END
BEGIN
mobule 9

1.5 Variables

For convenience, JACOBI 4 language has variables. Almost every module requires passing path to input and output files. Long paths might decrease redability of the script and result in hard-to-notice mistakes. The use of variables fixes this issue. It's worth mentioning that for files in current work directory its name could be used without path.

Consider the following example:

path	equals	data/in		
file	:=	work.csv		
c	equals	2		
transpose	≪path≫.csv	≪file≫		//data/in.csv
normalizeLength	≪file≫	n_≪file≫		//n_work.csv
log	n_≪file≫	l_≪file≫	≪c≫	//l_work.csv

To define a variable one should put its name in the first cell, keyword "assign", "equals" or ":=" in the second and value in the third.

To reference variable's value its name should be put in ≪≫. Variables could be used anywhere in the script to represent value or be part of value, but it is not possible to use variables instead of keywords.

It is also possible to assign value from a file using "from file" and supplying file name:

seed	:=	from file	next_seed.csv

Variable will be assigned to top left value of the input matrix.

1.6 Loops

JACOBI 4 has three types of loops:

- n-loop (repeat block n times).
- Iteration with variable.
- Loop over a set of values.

General considerations when using loops:

- Loops can be nested; there is no limit for nesting.
- Variables substituted at runtime. Issues related to incorrect variable's value would not be diagnosed prior to script execution.
- Variables could be accessed only in the scope (including nested ones) were defined.
- "BEGIN-END" blocks cannot intersect loops i.e. "BEGIN-END" block should be either inside the loop block or include ot in its entirety.

1.6.1. n-loop (repeat block n times)

The most simple loop in JACOBI 4 a loop for repeating block execution n times.

Consider the following earthbound example: add a hundred bootstrap copies of a given row. Lets assume in.csv contains given row.

copy	in.csv	result.csv	
Loop	100	times	
bootstrap	in.csv	b_copy.csv	
appendRight	result.csv	b_copy.csv	result.csv
end loop			

After script execution file result.csv will contain a row with its hundred bootstrap copies.

1.6.2. Iteration with variable

Consider the following example:

Loop	for	i	from	1	to	4	step	1
transpose	in≪i≫.csv	in≪i≫_t.csv						
end loop								

Variable "i" will held values 1, 2, 3, 4. The counting starts with value 1 (from | 1), at each step will be incremented by 1 (step | 1) until it reaches the value 4 (to | 4) including it.

The same result could be achived with the folloing script:

transpose	in1.csv	in1_t.csv
transpose	in2.csv	in2_t.csv
transpose	in3.csv	in3_t.csv
transpose	in4.csv	in4_t.csv

Loop variable adhere to the same rules as non-loop variables. Its scope is limited by the loop block.

1.6.3. Loop over a set of values

For this loop on each iteration valiable is assigned to a new value in a set.

Loop	for	i	from	in1	in2	matrA
transpose	≪i≫.csv	≪i≫_t.csv				
end loop						

The script execution will result in creating of files in1_t.csv, in2_t.csv and matrA_t.csv.

1.7 Module description file

Prior to script execution JACOBI 4 loads information about all available modules from module description files. It is used to validate script correctness and to translate module name to its executable.

Module description file is any ".csv" file with prefix "list" located in folder "JACOBI 4" or "JACOBI 4/bin"[2]. Examples of appropriate names: list.csv, LIST.ru.TXT, list_user.CSV.

Module description file contains table where each non-empty line contains description of one module:

internal	module	impl	string	integer:positive
internal	module2	impl2	string	string:optional
internal	echo	echo	any	
external	my_module	path/to/exe	count	double
subroutine	my_script	path/to/script.csv	integer	string

In this table the first column contains one of the following record type:

1. internal - internal module; path to the executable should be relative to the module description file.
2. external - external module; path to the executable will be resolved at runtime relative user selected working directory.
3. subroutine - JACOBI 4 script; path to the script should be relative to the module description file.
4. alias - special type that allows to set another name for a module.

The second column contains module name for use in scripts. Each module should have unique case-insensitive name. All duplicates will be ignored if there are ones.

For the alias type the third column contains the name of previously defined module. Name resolution happens after loading all module description files from a folder.

[2]Lets say JACOBI 4 is installed in D:\tools\JACOBI4. Then module description files would be loaded from D:\tools\JACOBI4\bin and D:\tools\JACOBI4

For the internal, external and subroutine types the third column contains path to the executable (or script); the fourth column and the following ones have a list of argument types for the module.

The following argument types are supported: string, integer, double, count, any. Types string, integer, double refer to values of the said type. count is used to pass a vector of values with type from the following column. any is used to define arbitrary set of arguments.

For each type it is possible to add attributes with the ":" symbol. Any type could have an "optional" attribute. For types integer and double attribute "nonnegative" could be added.

Chapter 2

Modules

2.1 Multivariate analysis

2.1.1. PCA (Principal Component Analysis)

Description

Principal component analysis is used as a dimentionality reduction tool.
Given the singular value decomposition of the input matrix X:

$$X = USV^T,$$

where S – diagonal matrix of singular values; U, V – orthogonal matrices. PCA could be expressed using SVD as follows:

$$T = US, \quad P = V,$$

where T is the score matrix, P is a loadings matrix.

Parameters

→ Input file name; contains matrix X.

← Output file name for scores matrix T.

← Output file name for loadings matrix P.

← Output file name for diagonal matrix of singular values S.

← (optional) Output file name for the vector of eigenvalues of input matrix X with cumulative statistics.

Example

BEGIN					
pca	in.csv	t.csv	p.csv	s.csv	
pca	in.csv	t.csv	p.csv	s.csv	vs.csv
END					

2.1.2. PCO (Principal Coordinates Analysis)

Description

Principal Coordinates is used for computing principle components matrix from euclidean distances matrix.

The module implements classical Gower algorithm (Gower, 1966):

1. Input matrix is squared, double cenetered and multiplied by $-\frac{1}{2}$.
2. SVD is used to calculate eigenvectors and diagonal matrix of singular values.
3. Negative singular values are zeroed. Square roots of singular values are computed.
4. Principal component matrix is computed by multiplying matrix of eigenvectors by the transformed matrix from the previous step.

Parameters

→ Input file name for table with euclidean distances matrix.

← Output file name for principal components matrix.

← Output file name for singular values.

← (optional) Output file name for vector of eigenvalues with cumulative statistics.

Example

BEGIN				
pco	in.csv	out.csv	s.csv	
pco	in.csv	out.csv	s.csv	vs.csv
END				

2.1.3. SVD (Singular Value Decomposition)

Description

Module computes singular value decomposition of input matrix X:

$$X = USV^T,$$

where S is a diagonal matrix of singular values; U, V – orthogonal matrices.

Parameters

→ Input file name.

← Output file name for matrix U.

← Output file name for matrix S.

← Output file name for matrix V.

← (optional) Output file name for vector of eigenvalues $\Lambda = S^2$ with cumulative statistics.

Example

BEGIN					
svd	in.csv	u.csv	s.csv	v.csv	
svd	in.csv	u.csv	s.csv	v.csv	vs.csv
END					

2.2 Metrics

2.2.1. Euclidean metric

Description

Module calculates euclidean distances between matrix rows:

$$d_{jk} = \sqrt{\sum_{i=1}^{I}(x_{ji} - x_{ki})^2}$$

For two input files module computes distances between corresponding rows of input matrices:

$$d_{jk} = \sqrt{\sum_{i=1}^{I}(x_{ji} - y_{ki})^2}$$

If x_{ji} or x_{ki} is empty the mean of squared difference of existing elements is used.

Parameters

→ Input file name.

→ (optional) The second file name.

← Output file name.

Example

BEGIN			
euclidean_metric	in.csv	out.csv	
euclidean_metric	in.csv	in2.csv	out.csv
END			

2.2.2. Minkowski metric

Description

Module calculates minkowski distances between matrix rows:

$$d_{jk} = \sqrt[r]{\sum_{i=1}^{I}(x_{ji} - x_{ki})^r}$$

For two input files module computes distances between corresponding rows from input matrices:

$$d_{jk} = \sqrt[r]{\sum_{i=1}^{I}(x_{ji} - y_{ki})^r}$$

If x_{ji} or x_{ki} is empty the mean of powered difference of existing elements is used.

Parameters

→ Input file name.

→ (optional) The second file name.

← Output file name.

→ Order r.

Example

BEGIN				
minkowski_metric	in.csv	out.csv	4	
minkowski_metric	in.csv	in2.csv	out.csv	4
END				

2.2.3. Jaccard index

Description

Module calculates binary similarity index for matrix rows:

$$K_J = \frac{c}{a + b - c}$$

where a is the amount of non-zero elements in the first vector, b - in the second vector, c is the amount of common elements in the first and second vectors.

For two input files module computes similarity index between corresponding rows from input matrices.

Parameters

→ Input file name.

→ (optional) The second file name.

← Output file name.

Example

BEGIN			
jaccard	in.csv	out.csv	
jaccard	in.csv	in2.csv	out.csv
END			

2.2.4. Jaccard-Steinhaus index

Description

Module calculates Jaccard-Steinhaus index[1] between matrix rows x_i and y_j as follows:

$$K_{ij} = \frac{\sum_{l=1}^{L} \min(x_{il}, y_{jl})}{\sum_{l=1}^{L} \max(x_{il}, y_{jl})}$$

where L - number of elements in a vector, K_{ij} - matrix element value at ij.

For two input files module computes Jaccard-Steinhaus index between corresponding rows from input matrices.

Parameters

→ Input file name.

→ (optional) The second file name.

← Output file name.

Example

BEGIN			
jaccardSteinhaus	in.csv	out.csv	
jaccardSteinhaus	in.csv	in2.csv	out.csv
jaccardRuzicka	in.csv	out.csv	
jaccardRuzicka	in.csv	in2.csv	out.csv
jaccardNaumov	in.csv	out.csv	
jaccardNaumov	in.csv	in2.csv	out.csv
END			

[1] Also known as Jaccard-Ruzicka, Jaccard-Naumov index.

2.2.5. Hamming distance

Description

Module calculates Hamming distance between matrix rows. The distance is defined as the number of positions at which the corresponding values are different.

For two input files module computes distances between corresponding rows from input matrices.

Parameters

→ Input file name.

→ (optional) The second file name.

← Output file name.

Example

BEGIN			
hamming	in.csv	out.csv	
hamming	in.csv	in2.csv	out.csv
END			

2.2.6. Manhattan distance

Description

Module calculates Manhattan distance between matrix rows:

$$d_{jk} = \sum_{i=1}^{I} |x_{ji} - x_{ki}|$$

For two input files module computes distances between corresponding rows from input matrices:

$$d_{jk} = \sum_{i=1}^{I} |x_{ji} - y_{ki}|$$

If x_{ji} or x_{ki} is empty the mean of absoluted difference of existing elements is used.

Parameters

→ Input file name.

→ (optional) The second file name.

← Output file name.

Example

BEGIN			
manhattan	in.csv	out.csv	
manhattan	in.csv	in2.csv	out.csv
END			

2.2.7. p-distance

Description

Module calculates distance between genetic sequences. The distance is defined as the number of positions at which the corresponding values are different, divided by the element count of vector.

Input matrix should have two columns, one with keys and one with genetic sequences.

Parameters

→ Input file name.

← Output file name.

Example

BEGIN		
p_distance	in.csv	out.csv
END		

Data example

in.csv

seq1	CAGACAGTCC
seq2	CACACTGCCA

out.csv

	o1	o2
o1	0	0.4
o2	0.4	0

2.2.8. Jukes-Cantor distance

Description

Module calculates distance between genetic sequences. Jukes-Cantor distance is similar to p-distance except it takes into account the probability of inversions:

$$d = -\frac{3}{4}\ln(1 - \frac{4}{3}p)$$

Input matrix should have two columns, one with keys and one with genetic sequences.

Parameters

→ Input file name.

← Output file name.

Example

BEGIN		
jukes_cantor	in.csv	out.csv
END		

Data example

in.csv

seq1	CAGACAGTCC
seq2	CACACTGCCA

out.csv

	o1	o2
o1	0	0.571605039035173
o2	0.571605039035173	0

2.2.9. Kimura two-parameter distance

Description

Module calculates distance between genetic sequences. Kimura distance is similar to Jukes-Cantor distance except it takes into account the probability of transitions and transversions:

$$d = -\frac{1}{2}\ln(1 - 2P - Q) - \frac{1}{4}\ln(1 - 2Q),$$

where P is proportion of sites that show transitional differences, Q is the proportion of sites that show transversional differences.

Parameters

→ Input file name.

← Output file name.

Example

BEGIN		
kimura_distance	in.csv	out.csv
END		

Data example

in.csv

seq1	CAGACAGTCC
seq2	CACACTGCCA

out.csv

	o1	o2
o1	0	0.575646273248511
o2	0.575646273248511	0

2.2.10. Spectrum distance

Description

Modules merges spectra - coordinate, intencity pairs - of different sizes from a set of input files.

Grid pitch h and half window w are calculated from input parameters:

$$h = \frac{X_{end} - X_{begin}}{N}$$
$$w_i = \left(K_s + i * \frac{K_e - K_s}{N} \right) * h$$

where i is a node index, K_s is a starting half window multiplier, K_e is an ending half window multiplier.

Minimal value X_{min} and maximum value X_{max} for the x-coordinate for each spectrum should meet the following constraints:

$$X_{begin} < X_{min} - w_0; \qquad X_{end} > X_{max} + w_0;$$
$$X_{begin} < X_{min} - w_{N-1}; \qquad X_{end} > X_{max} + w_{N-1}.$$

Algorithm

For each spectra the beginning and ending values of x-coordinate are predefined (the X_{begin} and X_{end} input parameters). For each grid node i, for each x_j - x-coordinates in the interval $[ih - w_i, ih + w_i]$ with non-zero signal intencity y_j, the inpact on the node i is calculated as follows:

$$z(i, x_j) = y_j * f\left(\frac{|ih - x_j|}{w_i} \right);$$
$$f(x) = 1 - 3x^2 + 2x^3.$$

The cumulative inpact z_i on the node i is calculated as $z_i = S\{z(i, x_j)\}$ for each x_j inside window $[ih - w_i, ih + w_i]$, where S is a merging function: sum, average or maximum.

Parameters

→ The number of input files F.

→ Input file name 1.

→ ...

→ Input file name F.

← Output file name.

→ X_{begin}.

→ X_{end}.

→ Nodes count N.

→ Starting half window multiplier K_s.

→ (optional) Ending half window multiplier K_e. The default value is K_s.

→ (optional) Mergind function S: average, sum, max. The default value is average.

Example

BEGIN									
spectrumDistance	1	i.csv	o.csv	1000	3000	10	1		
spectrumDistance	2	i.csv	i2.csv	o.csv	1000	3000	10	1	
spectrumDistance	2	i.csv	i2.csv	o.csv	1000	3000	10	1	max
END									

Data example

For input file in.csv:

in.csv

CON	spectrum1	vals	spectrum2	vals
1	2536.6	1012.14	1536.26	1422.06
2	2610.6	1277.94	1596.51	1348.26
3			1680.86	1440.43
4			1745.31	1493.17

And the following module invocation:

spectrumDistance	1	in.csv	out.csv	1000	3000	10	1

The h and w would have the following values:

$$h = 200,$$
$$w = 200.$$

The output file out.csv would be:

out.csv

in.csv	spectrum1	spectrum2
1000	0	0
1200	0	0
1400	0	171.233496556762
1600	0	906.552348547181
1800	0	867.621806348876
2000	0	0
2200	0	0
2400	240.643343664361	0
2600	1019.5239838512	0
2800	10.38868863324	0

2.2.11. Mantel test

Description

Module calculates Mantel test - a correlation between two matrices.

Parameters

→ Input file name for matrix A.

→ Input file name for matrix B.

← Output file name.

→ Number of iterations N.

Output file

The output file contains N, z, p-value and sum of squared deviations between according elements of input matrices.

variable	N	z	p-value	square error
value				

Example

BEGIN				
mantel_test	A.csv	B.csv	out.csv	1000000
END				

2.2.12. Rank Mantel test

Description

Module calculates rank Mantel test - a correlation between two matrices.

Parameters

→ Input file name for matrix A.

→ Input file name for matrix B.

← Output file name.

→ Number of iterations N.

Output file

The output file contains N, z, p-value and sum of squared deviations between according elements of input matrices.

variable	N	z	p-value	square error
value				

Example

BEGIN				
rank_mantel_test	A.csv	B.csv	out.csv	1000000
END				

2.2.13. Correlation

Description

Module calculates correlation between matrix rows of two input files.

Parameters

→ The first input file name.

→ The second input file name.

← Output file name.

→ (optional) Input matrix transposition options:

- t - transpose the first input matrix.
- nt - transpose the second input matrix.
- tt - transpose both the first and the second input matrices.

Example

BEGIN				
correlation	A.csv	B.csv	out.csv	
correlation	A.csv	B.csv	out.csv	tt
END				

2.2.14. Compression metric

Description

Module calculates objects similarity through compression ratio of textual object description:

combined_to_sum:

$$d_{jk} = 2\frac{c_{jk}}{c_{jj} + c_{kk}} - 1,$$

union:

$$d_{jk} = \frac{c_{jj} + c_{kk} - c_{jk}}{c_{jk}},$$

normalized_compression_distance:

$$d_{jk} = \frac{c_{jk} - min(c_{jj}, c_{kk})}{max(c_{jj}, c_{kk})},$$

where c_{jk}, c_{jj} and c_{kk} are sizes of compressed descriptions for objects j and k as if it was one string.

For the compress_metric module input file should consist of a column with textual description of objects along with row and column keys.

For the compress_metric_for_files module input file should consist of a column with file names along with row and column keys. Each file should contain textual description of one object.

Parameters

→ Input file name.

← Output file name for matrix $D = d_{jk}$.

← Output file name for matrix $O = c_{jj}$.

← Output file name for matrix $C = c_{jk}$.

→ Metric type: combined_to_sum, union or normalized_compression_distance.

Example

BEGIN					
compress_metric	in.csv	d.csv	o.csv	c.csv	union
compress_metric_for_files	files.csv	d.csv	o.csv	c.csv	union
END					

Data example

in.csv

c	data
o1	description for object 1
o2	data
o3	another description

files.csv

c	file names
o1	object_1.csv
o2	object_2.csv
o3	object_3.csv

d.csv

c	o1	o2	o3
o1	1	0.233333333333333	0.432432432432432
o2	0.233333333333333	1	0.28
o3	0.472222222222222	0.28	1

o.csv

c	compressed size
o1	29
o2	8
o3	24

c.csv

c	o1	o2	o3
o1	29	30	37
o2	30	8	25
o3	36	25	24

2.3 Matrix operations

2.3.1. Centering

Description

Module performs matrix rows centering: row mean is subtracted from each value in a row.

Parameters

→ Input file name.

← Output file name.

→ (optional) Input matrix transposition options:

- t - transpose input matrix.

Example

BEGIN			
centre	in.csv	out.csv	
centre	in.csv	out.csv	t
END			

2.3.2. Double centring

Description

Module performs double centring:

$$g_{ij} = a_{ij} - \overline{a_i} - \overline{a_j} + \overline{a},$$

where $\overline{a_i}$ - row mean, $\overline{a_j}$ - column mean, $\overline{a}$ - mean over all matrix elements $A = a_{ij}$.

Parameters

→ Input file name.

← Output file name.

→ (optional) Input matrix transposition options:

- t - transpose input matrix.

Example

BEGIN			
centreDouble	in.csv	out.csv	
centreDouble	in.csv	out.csv	t
END			

2.3.3. Normalize length

Description

Module normalized length of each matrix row by dividing its elements by the row length s_j:

$$s_j = \sqrt{\sum_{i=1}^{I} x_{ij}^2},$$

where I is a input matrix column count.

Parameters

→ Input file name.

← Output file name.

→ (optional) Input matrix transposition options:

- t - transpose the first input matrix.

Example

BEGIN			
normalizeLength	in.csv	out.csv	
normalizeLength	in.csv	out.csv	t
END			

2.3.4. Matrix on elements operations

Description

Module applies a function to each elemet of input matrix. Each non-number function application result is replaced with -1.

Arithmetic functions:

add <x>, sub <x>, mul <x>, div <x>.

Trigonometric functions:

cos, sin, tg, ctg, arccos, arcsin, arctg, arcctg.

Hyperbolic functions:

cosh, sinh, tgh, ctgh, arccosh, arcsinh, arctgh, arcctgh.

Exponential and logarithmic functions:

exp - computes exponential function e^x;

log <base> - computes logarithm with specified base;

exp2 - computes binary exponent: 2^x.

Power functions:

pow <power> - raises value to the specified power;

sqrt - computes square root;

cbrt - computes cubic root.

Error and gamma functions:

erf - computes error function;

erfc - computes complementary error function;

gamma - computes gamma funciton;

lgamma - computes the natural logarithm of the absolute value of the gamma function.

Rounding functions:

ceil - computes the smallest integer value not less than value;

floor - computes the largest integer value not greater than value;

mod <x> - computes remainder of division by x;

trunc - computes the nearest integer not greater in magnitude than value;

round - computes the nearest integer value.

Comparison functions:

isgreater <x>, isgreaterequal <x>;

isless <x>, islessequal <x>;

islessgreater <x>, isequal <x>;

isunordered - checks whether the value is not a number.

Other functions:

abs - computes absolute value;

max <x> - computes maximum with x;

min <x> - computes minimum with x;

sg, sign - computes value sign.

entropy - computes $-v * ln(v)$ for each matrix element v.

Parameters

→ Input file name.

← Output file name.

→ Function name.

→ Function parameter when necessary.

Example

BEGIN				
matrixOnElementsOperation	in.csv	out.csv	sin	
matrixOnElementsOperation	in.csv	out.csv	pow	-1
END				

2.3.5. Compute statistics

Description

Module computes the following statistics for input matrix columns:

1. number of elements in the row;
2. mean;
3. standard deviation;
4. variance;
5. skewness;
6. excess kurtosis;
7. sum of elements;
8. sum of squares;
9. min;
10. max.
11. confidence interval min.
12. confidence interval max.

Parameters

→ Input file name.

← Output file name.

→ (optional) Significance level, 0.05 if not set.

Example

BEGIN			
stat	in.csv	out.csv	
stat	in.csv	out.csv	0.05
END			

2.4 Data preparation

2.4.1. Transpose

Description

Modules performs matrix transposition.

Parameters

→ Input file name.

← Output file name.

Example

BEGIN		
transpose	in.csv	out.csv
END		

2.4.2. Table row selection

Description

Module allows selecting rows from input table.

Semantic query description

Each query is enclosed into one of the following types of brackets: [query] - to query value that should be included;]query[- to query values that should be excluded.

There are four query types:

1. column template, e.g. for query [label3] all rows with key 'label3' will be included;
2. lexicographical enumeratio, e.g. for query [label1:label3] all rows with key lexicographically between "labe1" and "label3" will be included;
3. absolute value enumeration, e.g. for [label1..label3] all rows starting from the first row with key "label1" and ending with the first row with key "label3" after "label1" will be included. For the case of key duplication it is possible to specify label index with "#" (use negative index to count from the end), e.g. for query [label1#4..label3#-2] the first three "label1" keys will be skipped and for "label3" the last one is skipped, making a range from the fourth "label1" to the second to last "label3".
4. absolute row index using numerical (e.g. [$3..$-4]) or alphabetical (e.g. [$C..$AF]) reference.

Parameters

→ Input file name.

← Output file name for rows that satisfy query.

← Output file name for rows that doesn't satisfy query.

→ Query.

Example

BEGIN				
copyRows	in.csv	out.included.csv	out.excluded.csv	[$B..$AR]
END				

Query examples

A			B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
			1	2	a	b	c	2	4	6	b	e	d	c		3.1	a	e	1	d
[b:d]			0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	1
[b:d#2]			0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	1

[b:d#-2]	0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	0	1
[b:3.14]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[b:$9]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[b:$H]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[b:~]	0	0	0	1	1	0	0	0	1	1	1	1	0	0	0	1	0	1
[b:]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[b..d]	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
[b..d#2]	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[b..d#-2]	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
[b..3.14]	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
[b..$9]	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
[b..$H]	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
[b..~]	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[b..]	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
[2:e]	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
[2:e#2]	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
[2:e#-2]	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
[2:4]	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0
[2:$9]	0	1	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0
[2:$H]	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
[2:~]	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
[2:]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[2..e]	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
[2..e#2]	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
[2..e#-2]	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
[2..4]	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
[2..$9]	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
[2..$H]	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
[2..~]	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[2..]	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
[$5:c]	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0
[$5:c#2]	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0
[$5:c#-2]	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0
[$5:4]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[$5:$7]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[$5:$H]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[$5:~]	0	0	0	1	1	0	0	0	1	1	1	1	0	0	0	1	0	1
[$5:]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[$5..c]	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
[$5..c#2]	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
[$5..c#-2]	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
[$5..4]	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
[$5..$7]	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
[$5..$H]	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
[$5..~]	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[$5..]	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

[~:b]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[~:b#2]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[~:b#-2]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[~:4]	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	0
[~:$5]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[~:$H]	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
[~:~]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[~:]	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
[~..b]	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[~..b#2]	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
[~..b#-2]	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[~..4]	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
[~..$5]	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[~..$H]	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
[~..~]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[~..]	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
[:b]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[:b#2]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[:b#-2]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[:4]	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	0
[:$5]	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	0	1	0
[:$H]	1	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
[:~]	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
[:]	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
[..b]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
[..4]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
[..~]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
[..]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

2.4.3. Table column selection

Description

Module allows selecting columns from input table.

Semantic query description

Documentation for "Table row selection" module contains semantic query description in section "Semantic query description".

Parameters

→ Input file name.

← Output file name for columns that satisfy query.

← Output file name for columns that doesn't satisfy query.

→ Query.

Example

BEGIN				
copyColumns	in.csv	out.included.csv	out.excluded.csv	[$B..$AR]
END				

2.4.4. Append matrix to the right

Description

Module merges two tables by appending the second table to the right of the first one without taking into account row keys. If the second table contains more rows, the missing row keys will be filled using values from the second table.

Parameters

→ Input file name for the first table A_1.

→ Input file name for the second table A_2.

← Output file name.

Example

BEGIN			
appendRight	A1.csv	A2.csv	out.csv
END			

Data example

A1.csv

A	c1	c2	c3
r1	1	2	a
r2	b	3	4

A2.csv

B	c4	c5
r3	5	6
r4	c	d
r5	7	e

out.csv

A	c1	c2	c3	c4	c5
r1	1	2	a	5	6
r2	b	3	4	c	d
r5				7	e

2.4.5. Append matrix to the bottom

Description

Module merges two tables by appending the second table to the bottom of the first one without taking into account column keys. If the second table contains more columns, the missing column keys will be filled using values from the second table.

Parameters

→ Input file name for the first table A_1.

→ Input file name for the second table A_2.

← Output file name.

Example

BEGIN			
appendBottom	A1.csv	A2.csv	out.csv
END			

Data example

A1.csv

A	c1	c2
r1	1	2
r2	3	str
r3	5	6

A2.csv

B	c3	c4	c5
r4	str2	8	a
r5	9	10	b
r6	11	12	c

out.csv

A	c1	c2	c5
r1	1	2	
r2	3	str	
r3	5	6	
r4	str2	8	a
r5	9	10	b
r6	11	12	c

2.4.6. Append file

Description

Module appends the second file to the first one.

Parameters

→ Input file name for the first table A_1.

→ Input file name for the second table A_2.

← Output file name.

Example

BEGIN			
appendFile	a.csv	b.csv	out.csv
END			

Data example

A.csv

A	c1	c2
r1	1	2
r2	3	str
r3	5	6

B.csv

B	c3	c4	c5
r4	str2	8	a
r5	9	10	b
r6	11	12	c

out.csv

A	c1	c2	
r1	1	2	
r2	3	str	
r3	5	6	
B	c3	c4	c5
r4	str2	8	a
r5	9	10	b
r6	11	12	c

2.4.7. Make binary matrix

Description

Module builds binary matrix for each set of N columns based on data values used in the set.

Parameters

→ Input file name.

← Output file name.

→ Block size N.

Example

BEGIN			
makeBinMatrix	in.csv	out.csv	3
END			

Data example

in.csv

IN	p1	p2	p3	p4	p5	p6
o1	3	4	1	2	5	5
o2	1	2	1	1	1	2
o3	1	1	1	1	3	4

out.csv

IN	p1_1	p1_2	p1_3	p1_4	p4_1	p4_2	p4_3	p4_4	p4_5
o1	1	0	1	1	0	1	0	0	2
o2	2	1	0	0	2	1	0	0	0
o3	3	0	0	0	1	0	1	1	0

2.4.8. Matrix shifting

Description

Module shifts matrix down by specified number.

Parameters

→ Input file name.

← Output file name.

→ Shift depth.

→ The number of shifts.

Example

BEGIN				
matrixShifter	in.csv	out.csv	1	3
END				

Data example

in.csv

CON	p1	p2
o1	1	2
o2	3	str
o3	5	6

out.csv

CON	p1	p2	p1	p2	p1	p2
o1	1	2				
o2	3	str	1	2		
o3	5	6	3	str	1	2
o1			5	6	3	str
o2					5	6

2.4.9. bootstrap

Description

Bootstrap module generates new sample from the source one.

Parameters

→ Input file name.

← Output file name.

→ (optional) Random number generator seed.

← (optional) Output file name to write random value[2] to.

Example

BEGIN				NULL
bootstrap	in.csv	out.csv		NULL
bootstrap	in.csv	out.csv	123	NULL
bootstrap	in.csv	out.csv	123	next_seed.csv
END				NULL

Data example

in.csv

IN	c1	c2	c3
r1	1	2	3
r2	a	b	c
r3	3	4	5
r4	4	5	6
r5	5	d	e

out.csv

IN	c1	c2	c3
r3	3	4	5
r5	5	d	e
r3	3	4	5
r2	a	b	c
r4	4	5	6

[2] Random value could be used as a seed for the next invocation of this module.

2.4.10. bootstrap labels

Description

Module generates integer weights for row labels such that its sum equals to the number of rows.

Parameters

→ Input file name.

← Output file name.

→ Repeat count.

→ (optional) Random number generator seed.

← (optional) Output file name to write random value[3] to.

Example

BEGIN					
bootstrapLabels	in.csv	out.csv	5		
bootstrapLabels	in.csv	out.csv	5	123	
bootstrapLabels	in.csv	out.csv	5	123	next_seed.csv
END					

Data example

in.csv

IN	c1	c2	c3
r1	1	2	3
r2	a	b	c
r3	3	4	5
r4	4	5	6
r5	5	d	e

out.csv

IN	weight_1	weight_2	weight_3	weight_4	weight_5
r1	0	1	2	2	0
r2	1	2	3	0	1
r3	2	1	0	0	2
r4	1	1	0	1	1
r5	1	0	0	2	1

[3] Random value could be used as a seed for the next invocation of this module.

2.4.11. Row permutation

Description

Permutation module generates new sample from the input one by shuffling rows.

Parameters

→ Input file name.

← Output file name.

→ (optional) Random number generator seed.

← (optional) Output file name to write random value[4] to.

Example

BEGIN				NULL
permutation	in.csv	out.csv		NULL
permutation	in.csv	out.csv	123	NULL
permutation	in.csv	out.csv	123	next_seed.csv
END				NULL

Data example

in.csv

IN	c1	c2	c3
r1	1	2	3
r2	a	b	c
r3	3	4	5
r4	4	5	6
r5	5	d	e

out.csv

IN	c1	c2	c3
r3	3	4	5
r2	a	b	c
r5	5	d	e
r4	4	5	6
r1	1	2	3

[4] Random value could be used as a seed for the next invocation of this module.

2.5 File operations

2.5.1. Copy file

Description

Module makes a copy of input file.

Parameters

→ Input file name.

← Output file name.

Example

BEGIN		
copy	in.csv	out.csv
END		

2.5.2. Delete file

Description

Module removes a file. It is also possible to use wildcards e.g. in*.csv.

Parameters

→ File name to delete.

Example

BEGIN	
delete	in.csv
END	

Chapter 3

Methods

3.1 Principal Coordinates

The usual way of computing principal component consists of the following[1]: let X be centered matrix of object coordinates in some euclidean space. Applying singular value decomposition (SVD) gives:

$$X = PSV^T,$$

where P, V^T are orthogonal matrices, and S is a diagonal matrix of singular values of matrix X. Then matrix

$$U = XV = PS$$

is a matrix of principal components for X. It's possible to apply SVD to symmetric matrix XX^T:

$$XX^T = P\Lambda P^T,$$

where P is the same orthogonal matrix as for X, and Λ is a diagonal matrix of singular values for matrix XX^T. However

$$XX^T = PSV^T * VSP^T = PSSP^T = PS^2P^T.$$

Consequently,

$$S^2 = \Lambda, \quad S = \Lambda^{1/2}.$$

That is, the matrix of singular values for XX^T is the matrix of eigen values for X. That's why principal components should be computed as follows:

$$U = P\Lambda^{1/2}.$$

This makes a lot of practical sense when the number of objects is significantly less then the number of features, which is more and more common in biological research.

More than half a century ago, Gower (Gower, 1966) discovered that if we calculate the matrix D of Euclidean distances between rows X (previously centered and normalized

[1] Vadim M. Efimov, Kirill V. Efimov, Vera Yu. Kovaleva Principal component analysis and its generalizations for any type sequence (pca-seq) // bioRxiv 535112, doi: 10.1101/535112

by columns), square this distances, double center and multiply them by $-\frac{1}{2}$, then we get the XX^T matrix. Applying SVD to it we get the principal components. For this reason Gower called this method the principal coordinates (PCo) analysis. However, matrix X itself is not needed and may not even exist in numerical form. To calculate the principal components of a certain set of objects, it is enough to have a matrix of Euclidean distances obtained in any way. If we calculate matrix of Euclidean distances between the rows of the matrix of principal components, then it will coincide with the initial matrix of Euclidean distances D. This property can be used to verify the calculations.

PCo is quite often used for dissimilarity matrices, for which it is unknown whether they are Euclidean distances between objects or not. In the case of non-Euclidean distances, some diagonal values of the matrix Λ will be negative. Small negative diagonal values can sometimes arise due to the accumulation of computational errors. All such "components", as well as zero ones, should be excluded from consideration.

JACOBI 4 documentation

Vadim Efimov Denis Polunin Irina Shtaiger

October 25, 2020

Index

appendBottom, 38
appendFile, 39
appendRight, 37

bootstrap, 42
bootstrapLabels, 43

centre, 26
centreDouble, 27
compress_metric, 24
copy, 45
copyColumns, 36
copyRows, 33
correlation, 23

delete, 46

euclidean_metric, 10

hamming, 14

jaccard, 12
jaccardSteinhaus, 13
jukes_cantor, 17

kimura_distance, 18

makeBinMatrix, 40
manhattan, 15
mantel_test, 21
matrixOnElementsOperation, 29
matrixShifter, 41
minkowski_metric, 11

normalizeLength, 28

p_distance, 16
pca, 7
pco, 8
permutation, 44

rank_mantel_test, 22

spectrumDistance, 19
stat, 31
svd, 9

transpose, 32