df.module

`df.module`

df provides a set of tools for working with dataframes in Logo.

Requires csv.lg, st.lg str.lg and rs.lg

df.create

`df.create :types :list`

Creates a dataframe, defined as a list composed of two lists: a first list of data types ("char" for character variables, "num" for numeric variables) and a second one composed of sentences (flat lists of words) representing the variables in the dataframe, where in each sentence the first word is the variable name and the others its values.

Checks variable names and number, data types names and number, missing values. CR or LF characters are not allowed in dataframe's variables.

Parameters

types: List of data types

list: List of lists

Returns

A dataframe.

Notes

In UCBLogo, words (they would be called a character string in other languages) are one of the three kinds of information (arrays and lists the other two) that Logo can process (numbers are a special case of words). A list of words, such as [1 2 3], is called a sentence or flat list.

Missing values ("missing"): non numeric observations in a numeric variable and empty observations in a numeric or character variable.

Examples

make "df (df.create [num char num] (list se "id iseq 1 20 se "sex (rs.rep [female male] 10) se "age rs.rand 1 100 20))

df.print :df

make "df (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.print :df

df.check

`df.check :df`

Checks a dataframe (variables names and data types names and number, missing values).

Parameters

df: A possibly non valid dataframe

Returns

A dataframe.

Notes

Variables names should not be numbers.

Examples

show df.check list [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])

df.names

`df.names :df`

Returns a list of variable names

Parameters

df: dataframe

Examples

show df.names (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.variables

`df.variables :df`

Returns a list of variables

Parameters

df: dataframe

Examples

show df.variables (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.values

`df.values :df`

Returns a list of values

Parameters

df: dataframe

Examples

show df.values (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.types

`df.types :df`

Returns a list of data types

Parameters

df: dataframe

Example

show df.types (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.dim

`df.dim :df`

Returns a list of dimensions [row column]

Parameters

df: dataframe

Examples

show df.dim (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.variable

`df.variable :df :name`

Returns a single variable

Parameters

df: dataframe

name: variable name

Examples

show df.variable (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) "id

df.colnumber

`df.colnumber :df :name [n 1] [names firsts df.variables :df]`

Returns the column number of a variable

Parameters

df: dataframe

name: variable name

Examples

show df.colnumber (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) "id

df.variable.type

`df.variable.type :df :name`

Returns a variable type

Parameters

df: dataframe

name: variable name

Examples

```
show df.variable.type (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]) ) "id
```

df.variable.values

`df.variable.values :df :name`

Returns a list of values

Parameters

df: dataframe

name: variable name

Examples

```
show df.variable.values (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]) ) "id
```

df.delete.variables

`df.delete.variables :df :names`

Deletes variables in a dataframe

Parameters

df: dataframe

names: list of variables names

Returns

A dataframe.

Examples

```
df.print df.delete.variables (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]) ) [id age]
```

df.rename.variables

`df.rename.variables :df :names :newnames`

Renames variables in a dataframe

Parameters

df: dataframe

names: list of variables names

newnames: list of new variables names

Returns

A dataframe.

Examples

df.print df.rename.variables (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) [id age] [idnew agenew]

df.select.variables

`df.select.variables :df :names`

Selects variables in a dataframe

Parameters

df: dataframe

names: list of variables names

Returns

A dataframe.

Examples

df.print df.select.variables (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) [id age]

df.select.rows

`df.select.rows :df :name :ind`

Selects rows (observations) using a variable value

Parameters

df: dataframe

name: variable name

ind: variable value

Returns

A dataframe.

Examples

df.print df.select.rows (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) "sex "female

df.select.first.rows

`df.select.first.rows :df :n`

Selects first n rows (observations)

Parameters

df: dataframe

n: number of rows

Returns

A dataframe.

Examples

df.print df.select.first.rows (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) 2

df.select.last.rows

`df.select.last.rows :df :n`

Selects last n rows (observations)

Parameters

df: dataframe

n: number of rows

Returns

A dataframe.

Examples

```
df.print df.select.last.rows (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]) ) 2
```

df.drop.all.missing

`df.drop.all.missing :df`

Drops all rows with missing values

Parameters

df: dataframe

Returns

A dataframe.

Examples

```
df.print df.drop.all.missing (df.create [num char num] (list [id 1 2 missing 4] se "sex (rs.rep [female male] 2) [age 28 missing 35 40]) )
```

df.drop.missing

`df.drop.missing :df :names`

Drops rows with missing values in one or more variables

Parameters

df: dataframe

names: list of variables names

Returns

A dataframe.

Examples

```
df.print df.drop.missing (df.create [num char num] (list [id 1 2 missing 4] se "sex (rs.rep [female male] 2) [age 28 missing 35 40]) ) [age]
```

df.select.missing

`df.select.missing :df :names`

Selects rows with missing values in one or more variables

Parameters

df: dataframe

names: list of variables names

Returns

A dataframe.

Examples

df.print df.select.missing (df.create [num char num] (list [id 1 2 missing 4] se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) [id age]

df.struct

`df.struct :df`

Prints a dataframe description

Parameters

df: dataframe

Examples

df.struct (df.create [num char num] (list se "id iseq 1 20 se "sex (rs.rep [female male] 10) se "age iseq 20 39))

df.head

`df.head :df [n 10] [dec 2]`

Prints dataframe head

Parameters

df: dataframe

n: number of rows (default 10)

dec: number of decimals (default 1)

Examples

(df.head (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) 2)

df.tail

`df.tail :df [n 10] [dec 2]`

Prints dataframe tail

Parameters

df: dataframe

n: number of rows (default 10)

dec: number of decimals (default 1)

Examples

(df.tail (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40])) 2)

df.print

`df.print :df [dec 2]`

Prints dataframe

Parameters

df: dataframe

n: number of decimals (default 2)

Examples

df.print (df.create [num char num] (list se "id iseq 1 4 se "sex (rs.rep [female male] 2) [age 28 missing 35 40]))

df.cbind

`df.cbind :df1 :df2`

Combines two dataframe, with the same number of rows, by columns. Duplicated variable names are not allowed.

Parameters

df1: first dataframe

df2: second dataframe

Returns

A dataframe.

Examples

df.print df.cbind (df.create [num char] list [age 23 54 missing 31] se "sex (rs.rep [female male] 2)) (df.create [num] [[id 1 2 3 4]])

df.rbind

`df.rbind :df1 :df2`

Combines two dataframes, with the same variables, by rows.

Parameters

df1: first dataframe

df2: second dataframe

Returns

A dataframe.

Examples

df.print df.rbind (df.create [num] [[id 1 2 3 4]]) (df.create [num] [[id 5 6]])

df.left.join

`df.left.join :df1 :df2 :keys1 :keys2`

Combines two dataframes using join keys, the output contains rows for values of the keys that exist in the first (left) dataframe, whether or not those values exists in the second (right) dataframe.

Parameters

df1: first dataframe

df2: second dataframe

keys1: list of key variables in the first dataframe. No more than three, no duplicated combined values, no missing values

keys2: list of key variables in the second dataframe. No more than three, no duplicated combined values, no missing values

Returns

A dataframe.

Examples

```
make "a (df.create [num char] [[ID 20 40] [Name John Jane]] )

make "b (df.create [num char] [[ID 20 50] [Job Lawyer Doctor]] )

df.print df.left.join :a :b [ID] [ID]
```

df.inner.join

`df.inner.join :df1 :df2 :keys1 :keys2`

Combines two dataframes using join keys, the output contains rows for values of the keys that exist in both dataframes.

Parameters

df1: first dataframe

df2: second dataframe

keys1: list of key variables in the first dataframe. No more than three, no duplicated combined values,no missing values

keys2: list of key variables in the second dataframe. No more than three, no duplicated combined values, no missing values

Returns

A dataframe.

Examples

```
df.print df.inner.join :c :d [ID] [ID]
```

df.table.join

`df.table.join :df1 :df2 :keys1 :keys2`

Combines two dataframes using the second as a lookup table that should include all distinct combined values of the keys that exist in the first (left) dataframe. If not, the combined dataframe includes only rows corresponding to the matched lookup table keys.

Parameters

df1: first dataframe

df2: second dataframe

keys1: list of key variables in the first dataframe. No more than three, can have duplicated combined values, no missing values

keys2: list of key variables in the second dataframe. No more than three, no duplicated combined values, no missing values

Returns

A dataframe.

Examples

df.print df.table.join (df.create [num char] [[ID 1 2 1 2 2] [Sex m f f m m]]) (df.create [num char] [[ID 1 2] [N 2 3]]) [ID] [ID]

df.print df.table.join (df.create [num char] [[ID 1 2 1 2 2] [Sex m f f m m]]) (df.create [num char] [[ID 1] [N 2]]) [ID] [ID]

df.sort

`df.sort :df :names`

Returns a dataframe sorted by up to five keys.

Parameters

df: dataframe

names: list of sorting keys, no more than five.

Notes

Observations with missing values in the sorting keys are added to the end of the sorted dataframe.

Characters "(", ")" and ";" not allowed in the sorting keys.

Accented characters (UTF-8 Latin-1 Supplement) are not ordered correctly.

Examples

df.print df.sort (df.create [num char] list [id 1 4 2 3] se "sex (rs.rep [female male] 2)) [id]

df.print df.sort (df.create [char char] list [id bb aa èe dd] se "sex (rs.rep [female male] 2)) [id]

df.print df.sort (df.create [num char] list [id 1 4 missing 3] se "sex (rs.rep [female male] 2)) [id]

df.map

```
df.map :df :formula  :vars :newvar [type "num] [no.missing "true]
```

Computes and adds a new variable to a dataframe, using up to five variables

Parameters

df: dataframe

formula list, the formula used to compute the new variable

vars: list of variables names used in the formula, no more than five, default no missing values

newvar: new variable's name

type: new variable's type (default "num)

no.missing: no missing values in "vars"? (default "true)

Returns

A dataframe.

Notes

nota

Examples

make "c (df.create [char num] [[sex m f f f m m m m f f m f m m m] [val 10 5 8 24 21 12 17 19 18 30 20 1 3 7 23]])

make "d (df.create [char char num num] [[sex m f f m missing] [year 1920 1930 1930 1930 1950] [val 10 25 18 30 20] [dec 10.12 2.1 18.1 3.22 20.5467]])

df.print df.map :c [product ? ?] [val] "newvar

df.print df.map :d [quotient ?1 ?2] [year val] "newvar

df.print (df.map :c [ifelse and lessp ? 20 greaterp ? 10 ["true] ["false]] [val] "newvar "char)

df.filter

```
df.filter :df :formula  :vars  [type "char]
```

Returns a new dataframe containing a subset of the original dataframe, using up to five variables

Parameters

df: dataframe

formula list, the formula used to subset the original dataframe

vars: list of variables names used in the formula, no more than five, no missing values

type: new variable's type (default "char)

Returns

A dataframe.

Examples

```
make "c (df.create [char num] [[sex m f f f m m m m f f m f m m m]  [val 10 5 8 24 21 12 17 19 18 30 20 1 3 7 23]] )

df.print df.filter :c [ifelse and lessp ? 20 greaterp ? 10 ["true] ["false]] [val]
```

df.stack

`df.stack :df :names [type "char] [strict "false]`

Returns a dataframe reshaped from wide to long format. Measured variables are stacked in a variable named "variable", their values in a variable named "value".

Parameters

df: dataframe

names: list of identifier (id) variables.

type: data type of the resulting variable "value". Default "char".

strict: if "true" duplicated values in the combined values of the identifier variables or missing values in the identifier variables are not allowed. Default "false".

Returns

A dataframe.

Notes

The dataframe "df" should contains only identifier variables and measured variables (variables to be stacked).

If all of the measured variables types are numeric, use "num" for the data type of the resulting variable "value". otherwise "char" (the default).

Examples

```
make "iris (df.create [num num num num char] [[sepal.length 5.1 4.9 6.7 6.9] [sepal.width 3.5 3 3.1 3.1] [petal.length 1.4 1.4 5.6 5.1] [petal.width 0.2 0.2 2.4 2.3] [variety Setosa Setosa Virginica Virginica]] )

show :iris

show (df.stack :iris [variety] "num)

make "sales (df.create [num num num num] (list se "year (rs.rep [2001 2002 2003] 4) se "quarter (rs.rep  iseq 1 4 1 3) se "north iseq 1 12 se "south iseq 21 32) )

show :sales

show df.sort (df.stack :sales [year quarter] "num) [variable year quarter]
```

df.unstack

`df.unstack :df :names`

Returns a dataframe reshaped from long to wide format. The dataframe in long format should include only identifier variables and two variables named "variable" and "value".

Parameters

df: dataframe in long format.

names: list of identifier variabes.

Returns

A dataframe.

Notes

If the dataframe in long format is the result of applying "df.stack" to a wide dataframe, df.unstack recovers the data types of the original wide dataframe using a property list

Examples

make "iris (df.create [num num num num char] [[sepal.length 5.1 4.9 6.7 6.9] [sepal.width 3.5 3 3.1 3.1] [petal.length 1.4 1.4 5.6 5.1] [petal.width 0.2 0.2 2.4 2.3] [variety Setosa Setosa Virginica Virginica]])

show :iris

make "iris.long (df.stack :iris [variety] "num)

show :iris.long

show df.unstack :iris.long [variety]

df.flatten.long

`df.flatten.long :df :id :name :sep`

Flattens composite values in a variable of varying lengths. creating a long dataframe.

Parameters

df: dataframe

id: list of identifier variables.

name: name of the variable to be flattened.

sep: values separator character, default ","

Returns

A dataframe.

Notes

Each composite value is separated in its components using the separator character without removing leading and trailing spaces

Examples

make "df df.create [char num char] [[id 1 2 3 4 5] [age 23 24 25 22 23] [temp 20,5,5,E1,7,13,19,9,6,20 18,8,16,11,23,E2,8,E2,E2,E2,90,70,40 19,24,E9,16,6,12,10,22 E2,E0,15,7,8,10,E1,24,17,13,6 14,8,E0,16,22,24,E1]]

df.print :df

make "df.long (df.flatten.long :df [id age] "temp ",)

df.print :df.long

make "df.long (df.map :df.long [ifelse numberp ? [?] ["missing]] [temp] "temperature "num)

df.print :df.long

df.print st.summary :df.long "temperature

df.flatten.wide

`df.flatten.wide :df :id :name :sep [no.missing "true]`

Flattens composite values in a variable creating a wide dataframe

Parameters

df: dataframe

id: name of the id variable.

name: name of the variable to be flattned.

sep: values separator character, default ","

no.missing: excludes missing values? default "true.

Returns

A dataframe.

Notes

Each composite value is separated in its components using the separator character without removing leading and trailing spaces

Examples

```
make "df1 df.create [num char] (list (se "id 1 2) (se "v "aa,bb "cc,aa,dd))

df.print :df1

df.print (df.flatten.wide :df1 [id] "v ", )
```