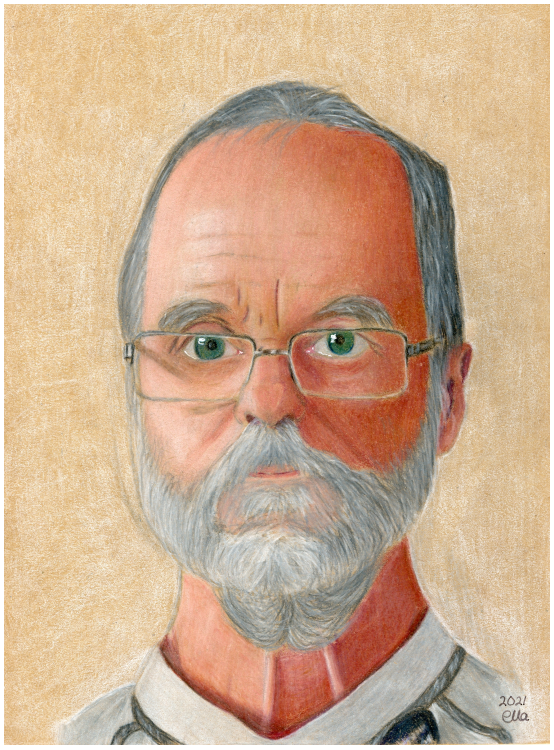
J
Users' Guide

Version 3.0

Juha Lappi
Reetta Lempinen

@date



Contents

1 find()

Function find() can be used to find the first matrix element satisfying a given condition. Remember that matrices are stored in row order. If a given column or row should be seaeched, use submatrix() to extract that row or column.

Args	1	Matrix	The matrix searched.
filter	1	Code	Gives the condition which the matrix element should be satisfied. The values of the matrix elements are put to the variable $.

Example 1 (findex)**.** *Example of find()*

```
 a=matrix(3,2,in->)
1,2
3,4
5,6
/
iel=find(a,filter->($.gt.4))
print(iel)
! <stat(data->xd)
```

```
 Accepted          1448
 var               min           max          mean            sd        sdmean
 d2             -144880.      507070.     -439.179       50065.4       1315.69
 d3             -16911.0      28687.0      315.262       2909.08       76.4489
 d4             -16911.0      40678.0      672.825       3719.46       97.7452
 d5             -16911.0      50702.0      1210.84       4995.30       131.274
 d6             -16911.0      61220.0      1903.98       6328.03       166.297
 i1             -107.200      16911.0      612.205       1650.59       43.3765
 one             1.00000      1.00000      1.00000       0.00000       0.00000
 <cpu1=cpu()
 <print('cpu',cpu1-cpu0)
 cpu
 =  0.31199999246746302
```

2 Predefined objects

The following objects are generated during the initilaization.

Names	Text	Text object containg the names of named objects
Pi	Var	The value of Pi (=3.1415926535897931)
Result	?	The default name of output object. Type varies according to the function

Arg	Var	The default argument name when using transformation object as a function.
Obs	Var	The default name of variable obtaining the the number of observation in a data set. Given in the data() function. Newdata ?
Record	Var	The name of variable obtaining the the number of record when reading data in data() function. Has the same value as Obs variabel if no records are rejected.
Subecord	Var	The name of variable obtaining the the number of record when reading subdata in data() function. Has the same value as obs variable if no records are rejected.
$Cursor$	Trans	The transformation object used to run sit> prompt
$Cursor2$	Trans	Another transformation object used to run sit> prompt
Val	Trans	Transformation object used to extract values of mathematical statements, used, e.g., in input programming.
$Data$	List	Default data set name for a new data set created by data()-function. call j_deflist3(0,'LastData',1,ivout_) ! ivlastdata) !!
LastaData	List	A list object referring to the last data set made, used as default data set. call j_getobject(0,'Data',j_ipreal,ivout_) ! ivcurrentdata) !!
Data	List	List object used to indicate current data setsDat
$	Var	Object name used to indicate console and '*' format in reading and writing commands.
x#	Var	Variable used when drawing functions.
Selected	Var	Variable used to indicate the simulator selected in simulations
Printinput	Var	Variable used to specify how input lines are printed. Not properly used.
Prinoutpu	Var	Variable used to indicate how much output is printed. Not properly used. call j_getobject(0,'Duplicate',j_ipreal,ivout_) ! ivduplicate) !!½!
Duplicate	Var	A special variable used in data() function when duplicating observations
$Buffer	Char	A special character object used by the write() function.
$Input$	Text	Text object used for original input line. and endcommnets are removed
1$Input1$	Text	Text object for input line after removing blanks and comments. $Input2$Text object for input line after interpreting "-sequencies.$
$Debug	Var	Variable used to put debugging mode on.
$Recursioion	Var	Variable telling recursion level of transformations.

$Crash	Var	Variable used to tell error handling system to crash the system when obtaining illegal values for some variables. This can be used to trace the calling sequence. of subroutines in case of *j* -errors. Not intended for users. $Warnings Variable telling the current number of printed warnings. If 30 warnings are printed then the printing of warnings is stopped. Printing of warnings can be reactivated by giving $Warnings=0 j_v(j_ivdac)=0 ! must be 4 for gfortran and 1 for intel Not needed, could be redefined
Maxnamed	Var	The maximum number of named objects. Determined via j.par in initilaization. default is 5000.
Round	Var	jlp(): The current round through treatment units in jlp() function. Can be used to define stopping criterion.
Change	Var	jlp(): The change of objective in jlp() in one round before finding feasible and thereafter the change in 10 rounds.
Change%	Var	jlp(): The change % change of objective in jlp() in one round before finding feasible and thereafter the change in 10 rounds.
Imp	Var	jlp(): The number of improvements obtained from schedules outside the current active set when updating the active set.
Testoptions	Var	By defining Testoptions=1, a developer can start to trace that options are cleared properly with j_clearoptions().
Accepted	Var	The number of accepted observations in functions using data sets.
$1	Var	Variable having value 1. Needed in regression context to indicate the regressor for intercept.
Continue	Var	If Continue has nonzero value then the control does not return to the sit> prompt when an error occurs, but computation proceed from the first ;incl() file. Used in the manual examples to demonstrate errors.
Err	Var	If Continue prevents the control from returning to sit> prompt this variable tells whether an error has occured.
In	Var	If several input files are indicated in in-> option in data() function the number of the input file is put into this variable and it can be used in transformations.

3 classify()

Classifies data with respect to one or two variables, get class frequencies, means and standard deviations of argument variables.

Output	1	Matrix	A matrix containing class information (details given below)
Args	1-	Var	Variables for which class means are computed.
data	N|1-	Data	data sets , see section Common options for default.
x	1	Var	The first variable defining classes. minobs minimum number of observation in a class, obtained by merging classes. Does not work if z-> is given
xrange	-1|0|2	Real	Defines the range of x variable. If xrange-> is given without arguments and J variables x%min and x%max exist, they are used, and if they do not exist an error occurs. Note that these variables can be generate with stat(min->,max->). If xrange-> is not given all values of the x variable define its own class.
dx	-1|1	Real	Defines the class width for a continuous x variable.
classes	-1|1	Real	Number of classes, If dx-> is not given, the default is that range is divided into 7 classes. minobs-> minimum number of observations in one class. Classes are merged so that this can be obtained. Does not work if z-> is present.
z	-1|1	Var	The second variable (z variable) defining classes in two dimensional classification.
zrange	-1|0|2	Real	Defines the range and class width for a continuous z variable. If J variables x%min and x%max exist, provided by stat(min->,max->), they are used.
dz	-1|1	Real	Defines the class width for a continuous z variable. mean if z variable is given, class means are stored in a matrix given in the mean-> option classes number of classes, has effect if dx is not defined in xrangedx->. The default is classes->7. If z is given then, there can be a second argument, which gives the number of classes for z, the default being 7.
trans	-1 | 1	Trans	transformation set which is executed for each observation. If there is a transformation set associated with the data set, those transformations are computed firs

filter	-1 | 1	Code	logical or arithmetic statement (nonzero value indicating True) describing which observations will be accepted. trans-> transformations are com
reject	-1 | 1	Code	logical or arithmetic statement (nonzero value indicating True) describing which observations will be rejected. trans-> transformations are com
print	-1|1	Real	By setting print->0, the classification matrix is not printed. The matrix can be utilized directly in drawclass() function.

Note 3.0.1. *If z variable is not given then first column in printed output and the first row in the outpututput matrix (if given) contains class means of the x variable. In the output matrix the last element is zero. Second column an TARKASTA VOISIKO VAIHTAArow shows number of observations in class, and the last element is the total number of observations. Third row shows the class means of the argument variable. The fourth row in the output matrix shows the class standard deviations, and the last element is the overall standard deviation*

Note 3.0.2. *Variable Accepted gives the number of accepted obsevations.ions.*

4 data()

Data objects are created with the data() function. Two linked data objects can be created with the same function call (using option subdata-> and options thereafter in the following description). It is recommended that two linked data objects are created with one data() function call only in case the data is read from a single file where subdata observations are stored immediately after the upper data observation. Data objects can be linked also afterwards with the linkdata() function. A data object can created by a data() function when data are read from files or data are created using transformation objects. New data objects can be created with newdata() function from previous data objects and/or matrices. If data objects can created using transformation objects either with data() function or by creating first data matrix by transformation and then using newdata() to create data object.

Output	0|1	Data	Data object to be created. If there is no output then the default is $Data$. It is recommended that this default is used only when only one data object is used in the analysis.
read	0|1-	Var|List	Variables read from the input files or the name of the list containing all variables to be read in. If no arguments are given and there is no readfirst-> option then the variables to read in are stored in the first line of the data file separated with commas.?? Also the ... -shortcut can be used to define the varaible list. If no arguments are given and there is readfirst-> option then the variable names are read from the second line.

in	0-	Char	input file or list of input files. If no files are given, data is read from the following input paragraph. If either of read-> or in-> option is given, then both options must be present.
form	-1|1	Char	Format of the data as follows

$	Fortran format '*'. The default
b	Single precison binary
bs	Single precison binary opened with access='stream'. Needed for Pascal files in Windows.
B	Double precison binary.
Char	General Fortran format, e.g. '(4f4.1,1x,f4.3)'
d4	Single precison direct access for Gfortran files.
d1	Single precison direct acces for Intel Fortran files.

maketrans	-1|1	Trans	Transformations computed for each observation when reading the data
readfirst	-1|0		variables read from the first line of the input file, if no variables are given and the first line starts with '!' or with '*' then the first line is printed (a text header). If no variables are given and the first line does not start with '!' or with '*' then the first line is a J-command line which is executed. The intended use for this is that the first line can contain both the names and values associated with the whole file. E.g the first line can be region,simulation_time=10,2016. ??
keep	-1|1-	Var	variables kept in the data object, default: all read-> variables plus the output variables of maketrans-> transformations.
obs	-1|1	Var	Variable which gets automatically the observation number when working with the data, variable is not stored in the data matrix, default: Obs. When working with hierarchical data it is reasonable to give obs variable for each data object.
filter	-1|1	Code	logical or arithmetic statement (nonzero value indicating True) describing which observations will be accepted to the data object. maketrans->-transformations are computed before using filter. Option filter-> can utilize automatically created variable Record which tells which input record has been just read. If observations are rejected, then the Obs-variable has as its value number of already accepted observations+1.

reject	-1|1	Code	Logical or arithmetic statement (nonzero value indicating True) describing which observations will be rejected from the data object. If filter-> option is given then reject statement is checked for observations which have passed the filter. Option reject-> can utilize automatically created variable Record which tells which input record has been just read. If observations are rejected, then the Obsvariable has as its value number of already accepted observations+1. subdata the name of the lower level data object to be created. This option is not allowed, if there are multiple input files defined in option in->.

subread,. . . ,subobs sub data options similar as read->. . . obs-> for the upper level data. (subform->'bgaya' is the format for the Gaya system). The following options can be used only if subdata-> is present

nobsw	-1|1	Var	A variable in the upper data telling how many subdata observations there is under each upper level observation, necessary if subdata-> option is present.
nobswcum	-1|1	Var	A variable telling the cumulative number of subdata observations up to the current upper data observation but not including it. This is useful when accessing the data matrix one upper level unit by time, i.e., the observation numbers within upper level observation are nobswcum+1,. . . ,nobswcum+nobsw
obsw	-1|1	Var	A variable in the subdata which automatically will get the number of observation within the current upper level observation, i.e. obsw variable gets values from 1 to the value of nobsw-variable, default is 'obs_variable%obsw'.
duplicate	-1|2	Trans	The two transformation object arguments describe how observations in the subdata will be duplicated. The first transformation object should have Duplicates as an output variable so that the value of Duplicates tells how many duplicates ar made (0= no duplication). The second transformation object defines how the values of subdata variables are determined for each duplicate. The number of duplicate is transmitted to the variable Duplicate. These transformations are called also when Duplicate=0. This means that when there is the duplicate-> option, then all transformations for the subdata can be defined in the duplicate transformation object, and submaketrans-> is not necessary.

oldsubobs	-1|1	Var	If there are duplications of sub-observations, then this option gives the variable into which the original observation number is put. This can be stored in the subdata by putting it into subkeep-> list, or, if subkeep-> option is not given then this variable is automatically put into the keep-> list of the subdata.
oldobsw	-1|1	Var	This works similarly with respect to the initial obsw variable as old-subobs-> works for initial obs variable.
nobs	-1|1	Real	There are two uses of this option. First, a data object can be created without reading from a file or from the following input paragraph by using nobs-> option and maketrans-> transformation, which can use Obs variable as argument. Creation of data object this way is indicated by the presence of nobs-> option and absence of in-> and read-> options. Second, if read-> option is present nobs-> option can be used to indicate how many records are read from a file and what will be the number of observations. Currently reject-> or filter-> can not be used to reject records (consult authors if this would be needed). If there are fewer records in file as given in nobs-> option, an error occurs. There are three reasons for using nobs-> option this way. First, one can read a small sample from a large file for testing purposes. Second, the reading is slightly faster as the data can be read directly into proper memory area without using linked buffers. Third, if the data file is so large that a virtual memory overflow occurs, then it may be possible to read data in as linked buffers are not needed. In case nobs-> option is present and read-> option is absent either maketrans-> or keep-> option (or both) is required.
buffersize	-1|1	Real	The number of observations put into one temporary working buffer. The default is 10000. Experimentation with different values of buffersize-> in huge data objects may result in more efficient buffersize-> than is the default (or perhaps not). Note that the buffers are not needed if number of observations is given in nobs->.
par	-1|1-	Real	additional parameters for reading. If subform-> option is 'bgaya' then par option can be given in form par->(ngvar,npvar) where ngvar is the number of nonperiodic x-variables and npvar is the number of period specific x-variables for each period. Default values are par->(8,93).
rfhead	-1|0		When reading data from a text file, the first line can contain a header which is printed but othewise ignored

rfcode	-1|0	The data file can contain also J-code which is first executed. Note the code can be like var1,var,x1...x5=1,2,3,4,5,6,7, which give the possibility to define variables which describe the in-> file. rfsubhead-> works for subdata similarly as rfhead-> for data. rfsubcode works for subdata similarly as rfcode-> for data If there are both rfhead-> and rfcode-> then rfhead-> is excuted first. rfhead-> and rfcode-> replace readfirst-> option of previous versions which was too complicated.

Note 4.0.1. *data() function will create a data object object, which is a compound object consisting-sisting of links to data matrix, etc. see Data object object. If Data is the output of the function, the function creates the list Data%keep telling the variables in the data and Data%matrix containg the data as a single precision matrix. The number of observations can be obtained by nobs(Data) or by nrows(Data%matrix).*

Note 4.0.2. *See common options section for how data objects used in other J functions will be defined.fined.*

Note 4.0.3. *The in-> and subin-> can refer to the same file, or if both are without argumentsuments then data are in the following input paragraph. In this case data() function read first one upper level record and then nobsw-> lower level records.*

Note 4.0.4. *When reading the data the obs->variable (default Obs) can be used in maketrans-etrans-> transformation and in reject-> option and filter-> option, and the variable refers to the number of observation in resulting data object. The variable Record gets the number of the read record in the input file, and can be used in maketrans-> transformations and in reject-> and filter-> options. If subdata-> option is given, variable Subreject gets the number of record in the sub file, and it can be used in submaketrans-> transformations and in subreject-> option and in subfilter-> option.*

Note 4.0.5. *Options nobs->100, reject->(Record.gt.100) and filter->ilter-> (Record.le.100) result in the same data object, but when reading a large file, the nobs-> option is faster as the whole input file is not read.*

Note 4.0.6. *If no observations are rejected, obs variable and Record variable get the same values.alues.*

Note 4.0.7. *If virtual memory overflow occurs, see nobs-> optio. This should not happen easily with the currrentrrent 64-bit application.*

Note 4.0.8. *Earlier versions contained trans-> and subtrans->options which associatediated a permanent transformation object with the data object. This feature is now deleted because it may confuse and is not really needed. If tranformations are needed in functions they can always be included using trans-> .*

5 Structure of general J functions

The general (non arithmetic) J functions are used either in statements

func(arg1,...,argn,opt1->value1,....,optm->valuem)
or
output=func(arg1,...,argn,opt1->value1,....,optm->valuem)

If there is no output for a function in a statement, then there can be three different cases: i) The function does not produce any output (if an output would be given, then J would just ignore it
ii) The function is producing output, and a default name is used for the output (e.g. Result for arithmetic and matrix operations, Figure in graphic functions).
iii) The function is a sub expression within a transformation consisting of several parts including other function or arithmetic operations. Then the output is put into a temporary unnamed object which is used by upper level functions as an argument (e.g. a=inverse(b)*t(c)) If the value of an option is not a single object or numeric constant, then it must be enclosed in parenthesis.
It is useful to think that options define additional argument sets for a function. Actually an alternative for options would be to have long argument lists where the position of an argument determines its interpretation. Hereafter generic term 'argument' may refer also to the value of an option.

When J is interpreting a function, it is checking that the option names and the syntax are valid, but it is not checking if an option is used by the function. Also when executing the function, the function is reacting to all options it recognizes but it does not notice if there are extra options, and these are thus just ignored.

An argument for a J function can be either functional statements producing a J object as its value, or a name of J object. Some options can be without any argument (indicating that the option is on).

```
a = sin(cos(c)+b)   ! Usual arithmetic functions have numeric values as arguments ! here the value of the argument of cos is obtained by 'computing' the value of real variable c.

stat(D,H,min->,max->)   ! Here arguments must be variable names

plotyx(H,D,xrange->(int(D%min,5), ceiling(D%max,5)))   !arguments of the function are variables, arguments of option xrange-> are numeric values

c = inverse(h+t(g))   ! The argument can be intermediate result from matrix computations. If it is evident if a function or option should have object names or values as their arguments, it is not indicated with a special notation. If the difference is emphasized, then the values are indicated by val1,...valn, and objects by obj1,...,objn, or the names of real variables are indicated by var1,...,varn. There are some special options which do not refer to object names or values. Some options define a small one-statement transformation to be used to compute something repeatedly.

stat(D,H,filter->(sin(D).gt.cos(H+1))   ! only those observations are accepted which pass the filter

draw(func->(sin($x)+1),x->$x,xrange->(0,10,1))   ! the func-> option transmits the function to be drawn not a single value.
```

6 Command input and output

J has two programming levels. First level, called input programming, generates text lines which are then transmitted to the interpreter which generates code which is the put into transformations sets or excuted directly. Input programming loops make it possible to generate large number of command lines in a compact and short form. This chapter describes input programming commands.

6.1 Input line and input paragraph

J reads input records from the current input channel which may be terminal, file or a text object. When J interprets input lines, spaces between limiters and function or object names are not significant. In input programming, functions start with ';' which is part of the function name (and there can thus be no space immediately after ';'). If a line (record) ends with ',' ,'+', '* ´, '-', '(', '=' or with '>', then the next record is interpreted as a continuation record and the continuation character is kept as a part of the input line. If a line ends with '»', then the nex line is also continuation line, and '»' is ignored. All continuation records together form one input line. In previous version input programming functions operated on input lines but now they operate on recors. One input record can contain 4096 characters, and an input line can contain also 4096 characters (this can be increased if needed). The continuation line cannot start with '*' or '!' because these are reserved to indicate comments. Note: '/' (division)cannot be used as last character indicating the continuation of the line because it can be legal last character indicating the end of an input paragraph.

When entering input lines from the keyboard, the previous lines given from the keyboard can no more be accessed and edited using the arrow keys owing to MSYS2 MSYS environment used to build the exe-file. To copy text from the J window into the clipboard right-click the upper left icon, select Edit, and then select Mark. Next click and drag the cursor to select the text you want to copy and finally press Enter (or right-click the title bar, select Edit, and in the context menu click Copy). To paste text from the clipboard into the J command line right-click the title bar, select Edit, and in the context menu click Paste. Console applications of Intel Fortran do not provide copy and paste using <cntrl>c and <cntrl>v. An annoying feature of the current command window is that it is possible All input lines starting with '*' will be comments, and in each line text starting with '!' will also be interpreted as comment (!debug will put a debugging mode on for interpretation of the line, but this debug information can be understood only by the author). If a comment line starts with '*!', it will be printed.

Many J functions interpreted and executed at the command level need or can use a group of text lines as input. In these cases the additional input lines are immediately after the function. This group of lines is called input paragraph. The input paragraph ends with '/', except the input paragraph of text function ends with '//' as a text object can contain ordinary input paragraphs. It may be default for the function that there is input paragraph following. When it is not a default, then the existence of the input paragraph is indicated with option in-> without any value. An input paragraph can contain input programming commands; the resulting text lines are transmitted to the J function which interprets the input paragraph

Example 2 (Inputparagraph)**.** *ph*

```
 tr=trans()
a=log(b)
write($,'(~sinlog is=~,f4.0)',sin(a))
```

```
/
b=matrix(2,3,in->)
1,2,3
5,6,7
/
```

6.2 Input programming

The purpose of the input programming is to read or generate J commands or input lines needed by J functions. The names of input programming commands start with semicolon ';'. There can be no space between ';' and the following input programming function. The syntax of input programming commands is the same as in J functions, but the input programming functions cannot have an out-put. There are also controls structures in the input programming. An input paragraph can contain input programming structures. 3.2.1. Addresses in input programming The included text files can contain addresses. Addresses define possible starting points for the inclusion or jump addresses within an include file. An address starts with semicolon (;) and ends with colon (:). There cannot be other text on the address line. E.g. ;ad1: See: ;incl, ;goto

Note 6.2.1. *The definition of a transformations set can also contain addresses. These addresses start start with a letter and end also with colon (:).*

6.2.1 ;incl

Includes lines from a file or from a text object.

Args	0|1	Ch|Tx	file name. Default: the same file is used as in the previous ;incl().
from	N|1	Ch	gives the starting in-> address for the inclusion, address is given without starting ';' and ending ':'.
wait	N|0		J waits until the include file can be opened. Useful in client server applications. See chapter J as a server.

Note 6.2.2. *Include files can be nested up to 4 levels.levels.*

Note 6.2.3. *See Chapter Defining a text object with text function and using it in ;incl how to includein-clude commands from a text object.*

Note 6.2.4. *When editing the include file with Notepad ++, it is reasonable to set the language as Fortran (free form). form).*

Example 3 (incltest)**.** *;incl('file.txt')*
;incl('file2.txt',from->'ad1')
;incl(from->'ad2')

6.3 Changing "..." sequences

If an original input line contains text within quotation marks, then the sequence will be replaced as follows. If a character variable is enclosed, then the value of the character variable is substituted: E.g. directory='D:\j\' name='area1 extension='svs' then in->'"directory""name"."extension"' is equivalent to in->'D:\j\area1.svs' If the "-expression is not a character variable then J interprets the sequence as an arithmetic expression and computes its value. Then the value is converted to character string and substituted into the place. E.g. if nper is variable having value 10, then lines

```
x#"nper+1"#"nper" = 56
chv = 'code"nper"'
```

are translated into

```
x#11#10 = 56
chv = 'code10'
```

With "..." substitution one can define general macros which will get specific interpretation by giving values for character and numeric parameters, and numeric parameters can be utilized in variable names or other character strings. In transformation sets one can shorten computation time by calculating values of expressions in the interpretation time instead of doing computations repeatedly. E.g. if there is in a data set transformation x3 = "sin(Pi/4)"*x5 Then evaluation of sin(Pi/4) is done immediately, and the value is transmitted to the transformation set as a real constant. If value of the expression within a "" sequence is an integer then the value is dropped in the place without the decimal point and without any spaces, otherwise its value is presented in form which is dependent on magnitude of the value. After J3.0 the format can be explicitly specified within [] before the numeric value. Eg. text can be put into a figure as fig = drawline(5,5,mark->'y="[f5.2]coef(reg,x1)"*x1+"[f5.2]coef(reg,1)"") See file jex.txt and Chapter 8 for an ex

7 Data functions

Data functions can generate data objects by reading data from files, or forming data objects by combing matrices and data objects. An important property of data objects is that they can be linked so that each observation of an upper level data is linked to several observations in lower level data. Each upper level observation can contain different number of lower level observations. Linear programming in forest management planning is based on linked data sets where each stand is connected to number of treatment schedules.

7.1 newdata()

Function newdata() generates a new data object from existing data objects and/or matrices possibly using transformations to generate new variables.

Output	1	Data	The data object generated.
Args	1-	Data|Matrix	Input matrices and data objects.

read	N|1-	Var	Variable names for columns of matrices in the order of matrices.
maketrans	N|1	Trans	A predefined ransformation object computed for each observation.

Note 7.1.1. *It is not yet possible to drop variables.bles.*

Note 7.1.2. *An error occurs if the same variable is several times in the variable list obtainedained by combining variables in data sets and read-> variables.*

Note 7.1.3. *An error occurs if the numbers of rows of matrices and observations in data sets sets are not compatible.*

Note 7.1.4. *Output variables in maketrans-> transformations whose name start with $ are not put into the new data object.ject.*

Example 4 (newdata)**.** *newdata() generates a new data object.*

```
 data1=data(read->(x1...x3),in->)
1,2,3
4,5,6
7,8,9
/
matrix1=matrix(3,2,in->)
10,20
30,40
50,60
/
newtr=trans()
;do(i,1,3)
;do(j,1,2)
x"i"#z"j"=x"i"*z"j"
;enddo
;enddo
/
new=newdata(data1,matrix1,read->(z1,z2),maketrans->newtr)
print(new)
```

7.2 exceldata()

Generates data object from csv data generated with excel. It is assumed that ';' is used as column separator, and first is the header line generated with excel and containing column names. The second line contains information for J how to read the data. First the first line is copied and pasted as the second line. To the beginning of the second line is put '@#'. Then each entry separated by ';' is edited as follows. If the column is just ignored, then put '!' to the beginning of the entry. If all characters in the column are read in as a numeric variable, change the name to accpetable variable name in J. If the column is read in but it is just used as an input variable fot maketrans-> trasformations, then start the name with '$' so the variable is not put to the list of keep-> variables. If a contains only character values then it must be ignored using '!'. If the contains numeric values

surrounded by characters, the the numeric value can be picked as follows. Put '?' to the end of entry. Put the variable name to the beginning of the entry. then put the the number of characters to be ignored by two digits, inserting aleading zero if needed. The given the length of the numeric field to be read in as a numeric value. For instance, if the header line in the excel file is

```
Block;Contract;Starting time;Name of municipality;Number of stem;Species code
```

and the first data line could be

```
MG_H100097362501;20111001;7.5.2021 9:37;Akaa;20;103;1;FI2_Spruce
```

then the second line before the first data line could be

```
##block0808?;!Contract;!Starting time;!Name of municipality;stem;species0201?
```

therafter the first observation would get values block=97362501,stem=1, and species=2.

If there are several input files, the header line of later input lines is ignored, and also if the second line of later files starts with '##', then it is ignored. if any later lines in any input files start with 'jcode:', then the code is computed. This way variables decribing the whole input file can be transmitted to the data. Currently jcode-output variables can be transmitted to data matrix only by using the as pseudo outputvariables in maketrans-transformations, e.g., filevar1=filevar1, if filevar1 is generated in jcode transformation. If there are several input files the file number is put into variable In before computing maketrans transformations and this variable is automatically stored in the data matrix.

Output	1	Data	Data object generated
in	1-	Char	Files to read in.
maketrans	N|1	trans	Transformations used to compute new variables to be stored in the data.

8 J transformations

Most operation commands affecting J objects can be entered directly at the command level or packed into transformation object. In both cases the syntax and working is the same. A command line can define arithmetic operations for real variables or matrices, or they can include functions which operate on other J objects. General J functions can have arithmetic statements in their arguments or in the option values. In some cases the arguments must be object names. In principle it is possible to combine several general J functions in the same operation command line, but there may not be any useful applications yet, and possibly some error conditions would be generated. Definition: A numeric function is a J function which returns a single real value. These functions can be used within other transformations similarly as ordinary arithmetic functions. E.g. weights() is a numeric function returning the number of schedules having nonzero weight in a JLP-solution. Then print(sqrt(weights())+Pi) is a legal transformation.

9 Statistical functions

There are several statistical functions which can be used to compute basic statistics linear and and nonlinear regression, class means, standard deviations and standard errors in one or two dimensional tables using data sets. There are also functions which can be used to compute statistics from matrices, but these are described in Section 11.1

9.1 stat()

Computes and prints basic statistics from data objects.

Output	0-1	Var	kokopo
Args	0-99	Var	variables for which the statistics are computed, the default is all variables in the data (all variables in the data matrix plus the output variables of the associated transformation object) and all output variables of the trans->
data	N|1-	Data	data sets , see section Common options for default.
data	-1,99	Data	data objects , see section Common options for default! weight gives the weight of each observations if weighted means and variances are computed. The weight c transformation or it can be a variable in the data object See 11 on page 19 for more details.
min	-1,99	Var	defines to which variables the minima are stored. If the value is character constant or character variable, then the name is formed by concatenating the character with the name of the argument variable. E.g. stat(x1,x2,min->'%pien') stores minimums into variables x1%pien and x2%pien. The default value for min is '%min'. If the values of the min-> option are variables, then the minima are stored into these variables.
max	-1,99	Var	maxima are stored, works as min->
mean	-1,99	Var	means are stored
var	-1,99	Var	variances are stored
sd	-1,99	Var	standard deviations are stored
sum	-1,99	Var	sums are stored, (note that sums are not printed automatically)
nobs	-1 | 1	Var	gives variable which will get the number of accepted observations, default is variable 'Nnobs'. If all observations are rejected due to filter->

trans	-1 | 1	Trans	transformation object which is executed for each observation. If there is a transformation object associated with the data object, those transformations are com
filter	-1 | 1	Code	logical or arithmetic statement (nonzero value indicating True) describing which observations will be accepted. trans-> transformations are co
reject	-1 | 1	Code	@
transafter	-1 | 1	Trans	transformation object which is executed for each observation which has passed the filter and is not rejected by the reject->-option.

Note 9.1.1. *1: stat() function prints min, max, means, sd and sd of the mean computedmputed as sd/sqrt(number of observations)*

Note 9.1.2. *2: If the value of a variable is greater than or equal to 1.7e19,.7e19, then that observation is rejected when computing statistics for that variable.*

Example 5 (stat)**.** *stat(area,data->cd,sum->bon20,filter->(site.ge.18.5))*
stat(ba,data->cd,weight->area)
*stat(vol,weight->(1/dbh***2))*

10 Analyzing classified data

The are several functions which can be used to analyze classified data. All these functions are described here, even if they belong Figure functions and statistical functions.

10.1 class()

Function class() computes the the class of given value when classifying values similarly as done in classify().

Output	1	Var	The class number.
Args	1	Real	The value whose class is determined.
xrange	2	Real	The range of values.
dx	N|1	Real	The class width.
classes	N|1	Real	The number of classes.

Note 10.1.1. *Either dx-> or classes-> must be given. If both are given, dx-> dominates.ates.*

Note 10.1.2. *If stat() is used earlier for variables including Var1 and1 and options min-> and max-> are present, then xrange->(Var1%min,Var1%max) is assumed.*

11 Common options

There are some options which are used in many J functions. Such options are e.g.

in	0|1-	Char	Indicates from where the data are read in. If there are no arguments, then the data are in the following input paragraph. If the values are character constants or a character variables, then data are read in from files those names.
data	N|1-	Data	data sets , see section Common options for default. All data sets will be treated logically as a single data set. If the function is using data sets, the daenta sets are given in data-> option. All data sets will be treated logically as a single data set. If a J function needs to access data, and the data-> option is not given then J uses default data which is determined as follows. If the user has defined an object list Data consisting of one or more data sets, then these will be used as the default data set. E.g. Data=list(dataa,datab) When a data set is created, it will automatically become the only element in LastData list. If the Data list has not been defined and there is no data-> option, then the LastData dataset will be used.
trans	-1 | 1	Trans	transformation set which is executed for each observation. If there is a transformation set associated with the data set, those transformations are computed firs In all functions which are using data sets, trans-> option defines a transformation set which is used in this function.
filter	-1 | 1	Code	logical or arithmetic statement (nonzero value indicating True) describing which observations will be accepted. trans-> transformations are com

Example 6 (comopt). *data1*

```
 tr=trans()
xy=x*y
/
stat(xy,trans->tr)
```

11.1 Matrix functions

There are matrix function which can create matrices (objects with type Matrix), take submatrices from larger matrices, make matrices from submatrices, compute statitics from matrices, solve linear equations, compute inverse and transpose of a matrix. Sums, differences and products of matrices are defined using ordinary arithmetic operations +,- and *. Arithmetic operations can combine scalars and matrices in the normal way. Now all matrices are in double precision. The data functions store data single precision matrices (type Matrix0) which cannot be used in matrix computations. Single precison data matrices can can be changed to double precision matrices simply by xmat=data1%matrix, where data1 is a data object.

11.2 find()

Function find() can be used to find the first matrix element satisfying a given condition. Remember that matrices are stored in row order. If a given column or row should be seaeched, use submatrix() to extract that row or column.

Args	1	Matrix	The matrix searched.
filter	1	Code	Gives the condition which the matrix element should be satisfied. The values of the matrix elements are put to the variable $.

Example 7 (findex)**.** *Example of find()*

```
a=matrix(3,2,in->)
1,2
3,4
5,6
/
iel=find(a,filter->($.gt.4))
print(iel)
! <stat(data->xd)
```

```
Accepted         1448
var               min          max          mean           sd         sdmean
d2             -144880.      507070.     -439.179      50065.4      1315.69
d3             -16911.0      28687.0      315.262      2909.08      76.4489
d4             -16911.0      40678.0      672.825      3719.46      97.7452
d5             -16911.0      50702.0      1210.84      4995.30      131.274
d6             -16911.0      61220.0      1903.98      6328.03      166.297
i1             -107.200      16911.0      612.205      1650.59      43.3765
one             1.00000      1.00000      1.00000      0.00000      0.00000
<cpu1=cpu()
<print('cpu',cpu1-cpu0)
cpu
=  0.31199999246746302
```