Appendix D

Summary list of IDL functions

This appendix lists the functions available in the IDL system, giving their arguments and a brief description of their behavior. The functions are described in the M-Lisp notation used in the Interlisp and IDL manuals. Thus, even though the function names appear here in lower-case, you must type them in upper-case to IDL.

adjoin[vector...]
: No-spread. Produces a new vector formed by joining its arguments together end to end.

anova[mtable;nesting]
: Produces a matrix containing a summary table for the analysis of variance of moments array **mtable**, where **nesting** is an optional specification of the nesting relations between the independent variables.

assign[target;source]
: The IDL assignment operator. **target** must be a selection on some array, and that array will be side-effected. The infix operator _ translates to *assign* if an array selection expression appears as its left-hand operand.

at[a;sltr]
: The IDL selection operator. Produces an array as output which is a window onto the section of **a** described by **sltr**. **sltr** may be a list of length no greater than the number of dimensions of **a** to specify a selection of **a**'s values, or a label or code selector. The infix operator @ is associated with *at*.

code[lev;val]
: Constructs a code selector to be given to *at*. If **lev** is NIL, the selector can be used to reference the value-labeled dimension of the array to which it is applied by *at*. Otherwise, if **val** is NIL, the selector will reference the complete codebook associated with level **lev** on the array's value-labeled dimension. If **val** is a non-NIL literal atom, it will reference the value associated with it on level **lev**. Finally, if **val** is a scalar, it will reference the label associated with that value.

copy[x]
: The Interlisp *copy* function extended to copy IDL arrays (which may be the value of **x** or may appear at arbitrary levels in the list-structure **x**). Assigning into a *copy* of an array will not affect the original.

counts[a]
: A generic function that returns the sum of the elements of **a** just like *rplus*, except that it skips over NIL's. In effect, it produces contingency tables for arrays that have been previously grouped.

covar[a;wt]
: Builds the covariation matrix (sums of mean centered cross products) for the columns of matrix **a**, using **wt** as a weighting vector. Returns a symmetric matrix, whose last row represents a mythical Constant variable (with value always 1) that has been swept out, so that the last row has the means of the columns of **a**, and -1/n in the last cell.

deal[n]
: Returns a vector containing a random permutation of the integers from 1 to **n**.

dumpidlarray[a;file]
: Writes a symbolic expression on **file** (primary output file if **file** is NIL) from which *readidlarray* can reconstruct an array equivalent to **a**. If **file** is not an open file, it is opened, the expression is printed, and then **file** is closed.

eapply[fn;expects;args]
: Extended APPLY. Applies **fn** to each section of the decomposition of **args** according to **expects**, an expectation list.

eapply*[fn;expects;args...]
: No-spread. Extended APPLY*. Applies **fn** to each section of the decomposition of **args** according to **expects**.

elementtype[a]
: Returns the type (INTEGER or FLOATING) of the elements of **a**, an array.

ems[nlevels;nesting;random]
: Produces an expected mean squares coefficient table for an analysis of variance design with factor levels given by **nlevels**, the nesting relationships by **nesting**, and the factors in **random** considered random. **nlevels** can be the moments table itself, as well as just the shape of its classification space.

extend[fn;expects]
: Modifies **fn** so that it will automatically extend across array arguments according to **expects**.

format[a]
: Returns the format (either FULL or SYMMETRIC) of array **a**.

fprob[f;dfnum;dfden]
: Returns the probability of *F*-value **f**, with degrees of freedom **dfnum**, **dfden**.

genvec[initial;end]
: Generates a vector of the numbers from **initial** to **end**. If **initial** is a two element vector, it is taken as the first two terms of the series (i.e. it specifies an increment). Thus, (GENVEC '(3 5) 11) generates the odd integers from 3 to 11.

group[attribs;values]
: Contructs an array of m+n dimensions, where m is the number of columns of the matrix **attribs**, and n the dimensionality of **values**. The values in each row of **attribs** are used as a subscript for an m-dimensional array whose extents are given by the number of distinct values found in the corresponding column of **attribs**. The output is formed by grouping all row-planes of **values** corresponding to equal rows of **attribs** in the m-space location addressed by that row. In effect, *group* places the row-planes of **values** within the cells of the classification design represented by **attribs**. If **values** is NIL, it is defaulted to a constant vector of 1's with length the number of rows of **attribs**. This produces an object from which *counts* will compute a contingency table.

hist[v;file]
: Prints a histogram for **v**, a vector, to the primary output file, or to **file** if given. Supresses multiple identical lines whenever their number exceeds the value of HISTRPTLINES. If **file** is not an open file, it is opened, the histogram is printed, and then **file** is closed.

idlmatrix[data]
: Produces a two-dimensional IDL array from **data**, a list structure describing its labels

and values. **data** is a list each element of which is a list representing one row of the output matrix: the CAR contains the row label to be assigned, and the CDR is a list of the values for that row. The row specifications may be optionally preceded by one list of the form (TITLES title-string dimension1-label dimension2-label), and another of the form (LABELS level1-labels level2-labels ...). The level labels are for levels on the second dimension of the output, and each one may be NIL (meaning no level label) a non-NIL literal atom which provides a level label but no codebook for the corresponding level, or a list of the form (level-label . codebook).

invert[m]
Produces the matrix inverse of **m**. Equivalent to -(SWEEP m ALL).

keep[ary;dims...]
No-spread. Produces a copy of **ary** with **dims** added to its kept dimensions. If **dims** is the literal ALL, then all the dimensions of **ary** will be kept. If no **dims** are specified, a vector of the currently kept dimensions is returned.

label[dim;lev]
Constructs a label selector to be given to *at* that will reference the label for dimension **dim** (if **lev** is NIL) or the label for level **lev** on dimension **dim**.

leave[ary;dims...]
No-spread. Produces a new array from **ary** with **dims** removed from its kept dimensions. If the first of **dims** is the literal ALL, then all of its kept dimensions will be eliminated.

listmatrix[m]
Converts IDL matrix **m** into a list structure from which *idlmatrix* can reconstruct the original.

mprod[a;b]
Matrix product of **a** and **b**. If they are vectors, *mprod* coerces them in the obvious way.

max[scalar...]
No-spread. Returns the maximum value of its arguments.

min[scalar...]
No-spread. Returns the minimum value of its arguments.

moments[a;wt;m]
Returns a vector of length **m**+1 containing the number of non-NIL values (moment 0) and the first **m** moments of array **a**, using **wt** as a weighting array. The first moment is the mean, the second is the variance, the third is the skew, etc. If **m** is NIL, it is defaulted to 2 and the n, mean, and variance are computed.

norm[m]
Norms matrix **m** by dividing each entry by the square root of the product of its basis {diagonal} elements

nprob[z]
Returns the one-tailed probability of a normal deviate of size **z**.

order[v;cfn]
Returns a permutation vector which will order vector **v** by comparison function **cfn**. If **cfn** is not specified, a function which will sort it into ascending order is used.

pairn[a;wt]
: Returns a matrix of the pairwise N corresponding to a covariation matrix on **a**, using **wt** as a weighting vector.

plot[y;x;file]
: Plots the vector **y** against the vector **x** (1 thru length **y**, if not given) on the primary output file, or **file** if specified. The axes are scaled so that they are approximately the same size. This scaling is controlled by the global variable PLOT.AXIS.RATIO. If **file** is not an open file, it is opened, the plot is printed, and then **file** is closed.

plusp[x]
: A predicate which is T if the scalar **x** is a positive number.

pool[mtable]
: Collapses **mtable** into a 3-vector of the pooled cell N, cell means, and mean-centered sum of squares. Effectively removes factors from a moments array.

ppa[a;file]
: Prints the array **a** on the primary output file, or **file** if specified. If **file** is not an open file, it is opened, the array is printed, and then **file** is closed. The precison of numbers in the table is determined by the global variable PRECISION, a list specifying the number of digits to appear to the right and left of the decimal point. The initial setting is (4 3) so that numbers are printed in the format rrrr.lll. The global variable ROWLABELWIDTH is an integer determining the number of columns to leave for the labels on rows (initally 8).

randn[mean;stdev]
: Returns a single number randomly sampled from the normal distribution with mean **mean** (default 0) and standard deviation **stdev** (default 1).

rank[a]
: Ranks the array **a** in ascending order. Result will be integer unless floating is necessary to resolve ties. Cells containing NIL are ignored in computing the ranks, and the corresponding cells in the value will be NIL.

readidlarray[file]
: Constructs an IDL array from an expression read from **file** (primary input file if **file** is NIL). If **file** is not an open file, it is opened, the expression is read, and then **file** is closed. *dumpidlarray* produces an expression of the appropriate form.

reduce[a;fn;startval]
: Applies function **fn** left associatively to the elements of **a**. If **startval** is given, **fn** is first applied to **startval** and $\mathbf{a}_1$, otherwise to $\mathbf{a}_1$ and $\mathbf{a}_2$. Thus, (REDUCE ARY 'NCONC1 NIL) returns a list of the elements of ARY. This function distinguishes the situtation where **startval** is specified as NIL from the case where it is not specified at all: (REDUCE ARY 'NCONC1) will leave $\mathbf{a}_1$ out of its value.

reshape[a;newshape;newformat]
: Reshapes **a** to an array of shape **newshape** and format **newformat**. The elements of **a** are put into the new array in rowmajor order. If **newshape** is NIL, **a** is simply flattened into a vector. If **newformat** is NIL, it is defaulted to FULL.

rplus[a]
: Equivalent to reduce[**a**;function[plus]], but more convenient to type.

rtimes[a]
: Equivalent to reduce[**a**;function[times]], but more convenient to type.

seek[sought;vec]
: **sought** can be a vector or a one argument function. Returns the (ordered) vector of indices of elements of **vec** which are found in, or which satisfy, **sought**.

shape[a]
: Returns a vector giving the shape of array **a**.

sweep[m;outvars;invars]
: Sweeps out variance components of matrix **m** corresponding to **outvars** (a selector for the second dimension of **m**); then sweeps in components corresponding to **invars**.

title[]
: Returns a selector that can be given to *at* to reference the title of an array.

tprob[x;df]
: Returns the probability of a *t*-value of **x** with degrees of freedom **df**.

translate[in;table;default]
: Translates the scalar **in** according to the matrix **table**. If **table** has one column (or is a vector), the index of (the first) **in** in **table** is returned. If it has two columns, the second element of the row whose first element is **in** is returned. If three columns, the third element of the row such that **in** is between the first and second elements is returned. In the three-column case, a NIL in the first two columns only matches a NIL **in**; otherwise, a NIL in column 1 is interpreted as -# while a NIL in column 2 is interpreted as +#. If no match is found, **default** is returned if it is specified; otherwise the value is **in** itself.

transpose[ary;perm]
: Transposes **ary** by mapping each of its dimensions onto the dimension of the output given by the corresponding element of **perm**. Two dimensions that are mapped onto the same dimension of the output are represented by their joint diagonal. If **perm** is not given, the dimensions are reversed.

In addition to the above, the mixed-arithmetic operators *plus*, *times*, etc., and most of the other arithmetic functions in Interlisp have been extended to apply element-wise across arrays. The infix operators +,*,-,/ have been mapped onto the mixed-arithmetic operators instead of the integer operators as in standard Interlisp.

IDL implements a new kind of Lisp function, an ELAMBDA, to simplify the task of defining extended functions. The key-word ELAMBDA is used instead of LAMBDA in the function definition, and the size of the expected object can be associated with each argument name. For example,
[DEFINEQ (FOO (ELAMBDA ((M MATRIX) (V VECTOR)) (CONS M V]
is equivalent to
[DEFINEQ (FOO (LAMBDA (M V) (EAPPLY* (FUNCTION (LAMBDA (M V) (CONS M V))) '(2 1) M V].

The IDL system defines the file command (prettydefmacro) IDLARRAYS for dumping IDL arrays via *makefile* onto ordinary Lisp files in *load*-able format. (Use the ARRAYS command to dump regular Lisp arrays.)

IDL automatically opens a Lisp DRIBBLE file at the beginning of each session, which maintains a transcript of all interactions with the system. This file is named IDL.TYPESCRIPT on the connected directory. It is a temporary file, and will disappear when the user logs off Tenex; it must be explicitly copied to another file if the contents are to be preserved.

Finally, the facilities of <LISPUSERS>SHOW are included in IDL. Thus, the command SHOW

may be given to the INTERLISP executive to cause the value of the last expression typed in to be pretty-printed. If that value happens to be an IDL array, the function *ppa* will automatically be invoked. The variable IT may be used in an expression to refer to the value of the preceding expression. These two augmentations to the Lisp system greatly improve the convenience of IDL interactions. For more details, see <LISPUSERS>SHOW.TTY.