Appendix A

ILWIS objects

Note: This appendix replaces Appendix B.1 in the ILWIS 2.1 Reference Guide.

Data objects

Raster maps, polygon maps, segment maps, point maps, tables and columns are called data objects.

Raster maps
A raster map consists of **pixels** (picture elements) of a certain size, e.g. 20m × 20m. Pixels are either codified by IDs, class names, values, or colors; this is determined by the domain of the map. A raster map should have coordinates, that is a georeference. In ILWIS, most spatial operations are performed on raster maps.

To obtain a raster map:
- rasterize an existing point, segment or polygon map, or
- create a new raster map and edit it with the pixel editor,
- use a satellite image which is already a raster map, or
- scan a map or photograph and import it into ILWIS.

Polygon maps
A polygon map is a vector map containing closed **areas** and the boundaries making up the areas. Polygon maps can for example contain uniquely codified areas such as cadastral plots, or mapping units such as land use classes, geological units or soil units. The areas are either codified by IDs, class names or values; this is determined by the domain of the map. Further, a polygon map uses a certain coordinate system. Polygon maps are generally used as a stepping stone to raster maps.

To obtain a polygon map, you first have to create a new segment map and edit it with the segment editor (with or without digitizer); then polygonize these segments within the segment editor or with the Segments to Polygons operation.

Segment maps
A segment map is a vector map containing **lines** (for example roads, rivers or contour lines). Segments are either codified by IDs, class names or values (height map); this is determined by the domain of the map. Further, a segment map uses a certain coordinate system.

To obtain a segment map, you should create one and edit it with the segment editor (with or without digitizer).

Point maps
A point map contains **points**, for example water wells or sample points. Points are either identified by IDs, class names or values; this is determined by the domain of the map. Further, a point map uses a certain coordinate system.

To obtain a point map you should create one and edit it with the point editor (with or without digitizer).

Map lists
A map list is a set of raster maps, for example the bands of a satellite image. All raster maps in the map list must have the same georeference and the same domain.

A map list is used for:

- sampling and classification
- creating a color composite
- a principal components analysis, or
- present multi-temporal changes in maps as a slide show.

A map list can be created from the File menu in the Main window or while starting an operation which requires a map list as input. You can include as many maps in a map list as you like.

(Attribute) Table
A table consists of records and columns. A table is an attribute table when the table stores additional information on elements in a map; i.e. extra tabular data which relates to mapping units, points, segments, or polygons in maps.

Raster, polygon, segment and point maps of the domain type Class or ID can have attribute tables. The domain of the attribute table should be the same as the domain of the map to which it relates. An attribute table can be linked to a map or to a domain through the Properties of a map or a domain.

An attribute table can be edited when the table is displayed in a table window. When the table is linked to a map or to a domain, and the map is displayed in a map window, you can also double-click the units in the map.

Columns
A table consists of columns. You can perform calculations with columns using TabCalc.

Each column has a domain. A column with a value domain contains values, a column with a class domain contains class names, a column with domain ID contains IDs, etc. Columns can also have domain String; you can use this to type for instance descriptions.

If in an attribute table you have columns with class domains, or with user-defined value domains, you may consider to prepare a representation for these columns as well. When you open the map to which the attribute table is linked, you can directly display the map by one of its attributes.

Service objects

Domains, representations, georeferences, and coordinate systems are called service objects.

Domains
A domain is a set of possible values of a variable. In ILWIS, a domain is the set of possible IDs, class names, or values that can be used for instance in a map. All maps, tables and columns (data objects) have a domain; a domain is a service object for maps, tables and columns. One domain can be used for several data objects.

The four main types of domains are:

▪ ID	for data objects which contain *unique identifiers* (e.g. plot 104, plot 105)
▪ Class	for data objects that contain *classes* (e.g. soil units: clay, sandy loam)
▪ Value	for data objects that contain measurable*, calculated or interpolated values* (e.g. height, concentration)
▪ Image	for *satellite images* containing values between 0 and 255.

Maps using a class, value or the image domain can have a user-defined representation.

Representations
A representation generally defines the manner in which the classes of a map with a class domain or the values of a map with a value domain or the image domain should be represented on the screen and on a printer. A representation is a service object for a domain, i.e. a domain uses a certain representation. For maps, which have a specific meaning (e.g. land use classes or height values) and which need fixed colors, it is advisable to create a user-defined domain and a user-defined representation. Maps, which use the same domain, are by default displayed in the same colors.

Representation types:

- *Representation class*: for maps that have a class domain and for raster maps with a group domain or a picture domain. For each class in the domain, a representation class contains: colors for mapping units in raster maps; colors, hatching or patterns for polygons; colors, line types, line widths, etc. or equally spaced symbols for segments; colors, symbol type, symbol sizes, etc. for points.
- *Representation value* or *representation gradual*: for maps that have a value domain or the image domain. Such representations contain colors assigned to ranges of values in raster, polygon, or segment maps.

Georeferences

A georeference is a service object which stores the relation between rows and columns of a raster map (row,col) and coordinates (X,Y). A georeference is needed for raster maps. A georeference uses a coordinate system. It is advised that raster maps of the same area use the same georeference.

- For rasterized vector maps, which are usually North-oriented, you can use a *georef corners*.
- For other raster maps, for example satellite images, which are not North-oriented and in which the pixels do not represent exactly square areas on the ground, you can use a *georeference tiepoints*.
- To add coordinates to a scanned photograph while using a Digital Terrain Model (DTM), create a *georef direct linear*.
- To add coordinates to a scanned aerial photograph with fiducial marks for which you have DTM, create a *georef orthophoto*.
- To create three dimensional views of maps while using a Digital Elevation Model, create a *georef 3D*.

To combine raster maps with different georeferences, for instance in MapCalc or Cross, first use the Resample operation, so that all maps will use one georeference.

Coordinate systems

A coordinate system defines the possible XY- or LatLon-coordinates that can be used in your maps and thus stores information on the kind of coordinates you are using in your maps. You may for instance use user-defined coordinates, coordinates defined by a national standard or coordinates of a certain UTM zone. Point, segment and polygon maps always have a coordinate system. Raster maps have a georeference, which uses a coordinate system. A coordinate system is a service object for point, segment and polygon maps, and for georeferences of raster maps.

There are five main types of coordinate systems:

- *coordsys boundary only*: to define XY-coordinates for maps by only specifying the boundaries of your study area.
- *coordsys projection*: to define XY-coordinates for maps by specifying the boundaries of your study area and optionally projection information, ellipsoid information and/or datum information.
- *coordsys latlon*: to define LatLon-coordinates for maps by specifying the boundaries of your study area in Latitudes and Longitudes and optionally ellipsoid information and/or datum information.
- *coordsys formula*: when you obtained data which is using different XY-coordinates than the coordinate system of your project, and when you know the relation between the two coordinate systems.
- *coordsys tiepoints*: when you obtained data which is using different XY-coordinates than the coordinate system of your project, and when you do not know the relation between the two coordinate systems.

☞ It is advised to use one coordinate system for all your maps. In case you have data of different sources in different projections, it is advised to *transform* all data to one common coordinate system.

Special objects

Map views, histograms, sample sets, two-dimensional tables, matrices, filters, functions, scripts are called special objects.

Map views

A map view is a saved map window. When a map view is opened, the set of data and/or annotation layers that it contains is directly displayed.

A map view contains the names of data layers and/or annotation layers to be displayed in one map window. Also, the display options of the layers are stored; so the system knows the colors, widths etc. for the display of each layer. Further, the georeference is stored, meaning that when you save a map view when zoomed in on a part of the map, this zoomed area will be displayed when opening the map view later.

Histograms

A histogram is a special object, which lists frequency information on values, classes or IDs in a raster, polygon, segment or point map. A histogram is automatically calculated when displaying a value map with stretching; you can also use the Histogram operation. The values in a histogram are presented as a table; optionally a graph can be shown.
Summary information of a value histogram (mean, standard deviation, and intervals) can be viewed through the Properties dialog box of the histogram.

A **raster histogram** lists the number of pixels, the percentages and the areas per value, class or ID. If the input raster map uses a domain Value, also cumulative number of pixels and cumulative percentages are calculated.

A **polygon histogram** lists the number of polygons and the perimeter and area of polygons per class, ID, or value. If the input polygon map uses a Value domain, also the cumulative number of polygons, cumulative perimeters and cumulative areas are calculated.

A **segment histogram** lists the number of segments and their length per class, ID or value. If the input segment map uses a Value domain, also the cumulative number of segments and cumulative lengths are calculated.

A **point histogram** lists the number of points per class, ID or value. If the input point map uses a Value domain, also the cumulative number of points are calculated.

Sample sets

Prior to an image classification, sample pixels or training pixels have to be selected in a sample set. To create a sample set, first a map list and a domain have to be specified. Then, with sampling, assign **class names** to groups of pixels that are supposed to represent a known feature on the ground and that have similar spectral values in the maps in the map list.

A sample set contains:

- a reference to a map list, that is the set of images you want to classify in a later stage. The spectral values of the images in the map list, at the position of the training pixels provide the basis on which decisions are made in the classification. During sampling, these values can be inspected in the sample statistics of a certain class of training pixels, and can be visualized in feature spaces;
- a reference to a class domain, that is the collection of class names that you want to assign to your training pixels and that are the classes that you want to obtain from the classification. The representation of this domain determines in which colors the training pixels are displayed during sampling;
- a reference to a raster map which is automatically created and obtains the same name as the sample set. This so-called sample map contains the locations of the training pixels and the class names assigned to them.

When your graphics board is configured to use 256 colors, you can locate your training pixels on a background map, for instance a color composite. In case your graphics board is configured to use more than 256 colors, you will use an interactive color composite; then, a background map is not used.

Two-dimensional tables
A two-dimensional table is used to combine two raster maps with class or ID domain. It defines a new value for each possible combination of input classes or IDs.

A two-dimensional table view consists of rows, which represent the domain of one map and of columns, which represent the domain of another map. You have to assign a new value, class name or ID to the fields, which represent the combination of the domains. Then you have to apply the two-dimensional table on the command line of the Main window. The output raster contains the values, classes or IDs as entered in the two-dimensional table.

Matrices
A matrix is a 2-dimensional array of values. Matrices are calculated by the Principal Components operation and by the Factor Analysis operation. The Principal Components operation calculates a.o. a variance-covariance matrix and the Factor Analysis operation calculates a.o. a correlation matrix. The matrices can be shown.

In the properties dialog box of a matrix, some additional information of the matrix can be viewed namely the total variances in the output bands.

Filters
Filters are used in the Filter operation.

ILWIS offers you:

- a choice of six filter types: Linear filters, Rank order filters, Majority filters, Binary filters, a Pattern filter and a Standard deviation filter. Each filter type calculates results by a different method;
- a choice of about 30 standard filters;
- the possibility to create, edit and store your own Linear filters, for instance through the New Filter option in the Operation-list;
- the possibility to modify the Average filter, Majority filters, Median filters, the Pattern filter, Rank Order filters, and the Standard Deviation filter according to your wishes in the Filter : dialog box.

Functions

Functions can be used in all calculators in ILWIS: MapCalc, TabCalc, scripts and the pocket line calculator. Some 50 internal functions are available (see also MapCalc and TabCalc), but also user-defined functions can be created.

A user-defined function may contain any combination of operators and functions, and may use parameters representing maps and columns. Parameters in a user-defined function can have any name.

Scripts

A script is a sequence of ILWIS expressions. By creating a script, you can build a complete GIS or Remote Sensing analysis for your own research discipline. Scripts are equivalent to batch files in ILWIS version 1.4.

For more information about script syntax, see Appendices: Operators and functions in MapCalc and TabCalc, Appendices: ILWIS expressions and Appendices: ILWIS script language (syntax).

Annotation Text

An Annotation Text object, also called an annotation text layer, is designed to display and store multiple texts at *multiple positions*. While simple annotation types can only be stored by saving a map window as a map view, an Annotation Text object can be stored by itself. An Annotation Text object can be edited with the Annotation Text editor or in a table window.
When creating an Annotation text object, you can base the texts that will appear in the text object on an existing point, segment, or polygon map. If you do so, the text object will contain a text string (class name, ID or value) for each point, segment or polygon in the selected map.

In the Annotation Text editor, you can then easily insert more text items, change and refine the position of texts (move), make text duplicates, and specify fonts, font sizes, appearance (bold, italics, underline), colors, rotations, etc. for (multiple) selected texts.

Appendix B

ILWIS operations

Note: This appendix replaces Appendix C in the ILWIS 2.1 Reference Guide.

VISUALIZATION

Show Map
Show a map in a new map window.

Show Table
Show a table in a table window.

Show Map List as Color Composite
Display three images or other raster maps with a value domain, present in a map list as a color composite. You will show an Interactive color composite.

By using an *interactive* color composite, you can easily change intervals, select other bands, etc. The resulting color composite is displayed in a map window, which can be saved as a map view. Interactive color composites are very suitable to be used as a background during sampling or during screen digitizing.
Your graphics board needs to be configured to use more than 256 colors, for instance High Color 16-bit, or True Color 24-bit (see Display Settings in Windows' Control Panel).

A *permanent* color composite can always be created with the Color Composite operation on the Operations, Image Processing menu.

Show Map List as Slide Show
Show multiple raster maps, which are combined in a map list, one after the other in a map window at a user-specified rate. You will show a Slide Show. This visualization technique is designed to present multi-temporal changes in maps. All maps in the map list must use the same domain and the same georeference.

You can use a slide show:
- to present the temporal changes of maps with the same theme but of different years,
- to display a number of classified satellite images of different months or years,
- to display derived products from satellite images such as NDVI maps of different months,
- to display a number of 3D views one after the other.

Display 3D
With Display 3D, you can create and edit a georeference 3D in order to obtain a three dimensional view of one or more maps. A Digital Elevation Model (DEM) is required to create a georeference 3D. The DEM can then be displayed as 3D grid lines with or without a drape of a raster map. The georeference 3D can be edited with the Georeference 3D editor.
When finished editing the 3D view, you can add point, segment and/or polygon maps and/or annotation to improve the 3D view.

Apply 3D
The Apply 3D operation resamples an input raster map according to a georeference 3D. This enables you to permanently and quickly display the output raster map in three dimensions, i.e. as a 3D view.
You can create a georeference 3D with Display 3D or Apply 3D; a Digital Elevation Model is required and you have to specify 3D view parameters using the Georeference 3D editor.

RASTER OPERATIONS

Map Calculation
Map Calculation can be used to perform calculations with raster maps. Map calculation is used for the execution of most spatial analysis functions and modelling operations. It integrates spatial and tabular data. The program enables the user to perform overlay, retrieval operations, and queries. Type map calculation formulae on the command line of the Main Window.

The following operations can be executed:
- manipulation of one or more raster maps by performing arithmetical, relational, logical, conditional, exponential, logarithmic and other operations,
- creation of attribute maps from map-related tables with attribute data,
- classification of domain Value maps according to a domain Group,
- application of user-defined functions.

Attribute map of raster map
By creating an attribute map of a raster map, the class name or ID of each pixel in the original map is replaced by the value, class or ID found in a certain column in an attribute table.

A raster map using a Class or ID domain, can have extra attribute information on the classes or identifiers in the map. These attributes are stored in columns in an attribute table. The attribute table can be linked to the map to which it refers, or to the domain of the map. You can check whether an attribute table is linked to the raster map or to its domain through the Properties dialog box of the map or the domain.

Cross
The Cross operation performs an overlay of two raster maps. Pixels on the same positions in both maps are compared; the occurring combinations of class names,

identifiers or values of pixels in the first input map and those of pixels in the second input map are stored. These combinations give an output cross map and a cross table. The cross table also includes the number of pixels that occur for each combination.

Aggregate map
The Aggregate map operation aggregates blocks of input pixels by applying an aggregation function: Average, Count, Maximum, Median, Minimum, Predominant, Standard Deviation or Sum. The Aggregate Map operation either creates a new georeference in which each block of input pixels corresponds to one output pixel (group) or the output raster map uses the same georeference as the input map (no group).

Distance calculation
Distance calculation assigns to each pixel the smallest distance in meters towards user-specified source pixels, for example to schools, to roads etc. The output is called a distance map.
The input map for a distance calculation is called a source map: all pixels with a class name, ID, or value are regarded as source pixels, and distance values will be calculated for all pixels that are undefined. In the Distance calculation dialog box, a source map can be any raster map with a class domain or an ID domain. On the command line, you can also use raster maps with a value domain.

Inaccessible or less accessible areas can be indicated in a weight map. The weight factors in such a map represent the relative difficulty, a 'resistance', to surpass pixels. By using weight factors that are inversely proportional to the possible speed that can be obtained in different mapping units, a so-called travel time map can be calculated. Through a distance calculation, also a Thiessen map can be calculated. A weight map can be used, but is not obligatory.

Iteration
Iterations are a special type of map calculations. They are a successive repetition of a mathematical operation, using the result of one calculation as input for the next. These calculations are performed line by line, pixel by pixel and take place in all directions. When a calculation in one direction is finished (for instance from top to bottom), a rotation takes place for the calculation in the next direction. The calculation stops when there are no more differences between an output map compared to the previous output map, or when a certain number of iterations is reached as defined before. Iterations are often used in combination with neighbourhood operations.

Area numbering
Area numbering is a raster operation, which assigns unique identifiers to pixels with the same class names or values that are horizontally, vertically or diagonally connected. The output of the Area numbering operation is a map in which these connected areas are codified as Area 1, Area 2, etc. Further, an attribute table is created with the map, which contains the new Area IDs and the original class names, IDs or values.

Area numbering can be used to make a decision based on the area of individual groups of pixels (uniquely identified areas) instead of on the total area of all pixels with the same class name or value.

Sub-map of raster map
The Sub-map of raster map operation copies a rectangular part of a raster map into a new raster map. The user has to specify row and column numbers of the input map to indicate the part of the input map that should be copied into the new raster map.

Glue raster maps
The Glue raster maps operation glues or merges two or more georeferenced input raster maps into one output raster map. The output map then comprises the total area of all input maps. The domains of the input maps are merged when needed. A resampling is performed when needed.

With the Glue raster maps operation, you can thus merge two or more adjacent or partly overlapping raster maps (i.e. make a mosaic), or glue smaller raster maps onto a larger one.

Mirror Rotate
The Mirror/Rotate operation allows you to reflect a raster map in a horizontal, vertical, or diagonal line, to transpose the map's rows and columns, or to rotate a raster map 90°, 180°, 270° (clock-wise).

IMAGE PROCESSING

Filter
Filtering is a raster operation in which each pixel value in a raster map is replaced with a new value.

The new value is obtained by applying a certain function to each input pixel and its direct neighbours. These neighbours are usually the 8 adjacent pixels (in a 3×3 filter) or the 24 surrounding pixels (in a 5×5 filter). When you create your own filters, any odd sized matrix is allowed (5×1, 11×23, 25×25); the maximum user-defined filter size is 8000.
Filtering is for instance used to sharpen a satellite image, to detect line features, etc.

Stretch
The Stretch operation re-distributes values of an input map over a wider or narrower range of values in an output map. Stretching can for instance be used to enhance the contrast in your map when it is displayed. Two stretch methods are available: linear stretching and histogram equalization.

Slicing
The Slicing operation classifies ranges of values of an input raster map into classes of an output map. A domain Group should be created beforehand; it lists the upper value boundaries of the groups and the output class names.

To perform an interactive slicing, you can create a representation value for the input map and change value boundaries and colors of the representation value.

Color separation
The Color separation operation allows you to extract different 'bands' for instance from a scanned or digital color photo as if using color filters when taking the picture. After color extraction, you can perform the normal Image Processing operations like Filtering, Classification, etc. on these bands.

Maps that have a Picture domain or the (24 bit) `Color` domain store for each pixel three values: Red, Green and Blue. The Color separation operation allows you to retrieve for each pixel either the Red, Green or Blue value and store these in a separate map. You can also retrieve Yellow, Magenta, Cyan, combined Gray values, or Hue, Saturation or Intensity values for each pixel.

Color composite
A color composite is a combination of three raster bands. One band is displayed in shades of red, one in shades of green and one in shades of blue. Putting three bands together in one color composite map can give a better visual impression of the reality on the ground, than by displaying one band at a time. Examples of color composites are false color (or IR) images and 'natural color' images.
The Color Composite operation on the Operations, Image Processing menu creates a *permanent* color composite raster map.

To *interactively* create a color composite, choose Show MapList as Color Composite from the Operations, Visualization menu. Interactively created color composites can be stored by saving the map window as a map view.

Cluster
Clustering, or unsupervised classification, is a rather quick process in which image data is grouped into spectral clusters based on the statistical properties of all pixel values. It is an automated classification approach with a maximum of 4 input bands.

Sample
Sample is an interactive process of selecting training pixels in a sample set prior to an image classification.

In the sample set editor, select pixels that are characteristic for a certain type of a certain natural resource on the ground and that have similar spectral values in the maps in the map list, and assign a class name to them. The spectral values of these sampled pixels or training pixels provide the basis on which decisions are made during classification. These values can be inspected in the sample statistics of a certain class of training pixels and can be visualized in feature spaces. The result of Sampling is a filled sample set.

Classify
The Classify operation performs a multi-spectral image classification according to training pixels in a sample set (supervised classification).

The following classification methods can be used:
- Box classifier
- Minimum Distance to Mean classifier
- Minimum Mahalanobis Distance classifier
- Maximum Likelihood classifier

Resample
The Resample operation resamples a raster map from the map's current georeference to another target georeference. The coordinate of each output pixel is used to calculate a new value from close-by pixel values in the input map. Three resampling methods are available: nearest neighbour, bilinear interpolation, and bicubic interpolation.

In raster operations (e.g. MapCalc, Cross), all input raster maps must have the same georeference. Thus, prior to such operations, use Resample:
- *to combine raster maps from various sources*, when maps use different coordinate systems (projections) or different georeferences (pixel size): resample the maps to one common georeference;
- *to combine satellite imagery of different dates or resolutions*: create a georef tiepoints for each set of images, then resample the images preferably to a georef corners;
- *to combine satellite images with rasterized vector maps*: rasterize the vector data on the georef tiepoints of the satellite images, or, in case you prefer North-oriented raster maps, rasterize the vector maps with a georef corners, and resample the images with the georef tiepoints to this georef corners;
- *to combine scanned photographs with rasterized vector data*, or *to rectify scanned aerial photographs*: create a georef tiepoints, a georef direct linear or a georef orthophoto for the photo, then resample the photo to a georef corners.

STATISTICS

Histogram
The Histogram operation calculates the histogram of a raster, polygon, segment or point map. Histograms list frequency information on the values, classes, or IDs in your map. Results are presented in a table and optionally in a graph. Summary information of a histogram of a Value raster map can be viewed in the properties of the histogram (mean, standard deviation, and percentage intervals).

A **raster histogram** lists the number of pixels, the percentages and the areas per value, class or ID. If the input raster map uses a Value domain, also cumulative number of pixels and cumulative percentages are calculated.
A **polygon histogram** lists the number of polygons and the perimeter and area of polygons per class, ID, or value. If the input polygon map uses a Value domain, also

the cumulative number of polygons, cumulative perimeters and cumulative areas are calculated.
A **segment histogram** lists the number of segments and their length per class, ID or value. If the input segment map uses a Value domain, also the cumulative number of segments and cumulative lengths are calculated.
A **point histogram** lists the number of points per class, ID or value. If the input point map uses a Value domain, also the cumulative number of points are calculated.

Raster

Autocorrelation - Semivariance
The Autocorrelation - Semivariance operation calculates the autocorrelation and semivariance of a raster map. The autocorrelation of a raster map is generated by calculating the correlation between pixel values of a raster map and pixel values of the same raster map for different shifts (lags) in horizontal and vertical directions. The semivariance, a measure for the spatial variability of a raster map, is calculated for the same shifts.

Map list

Principal Components
The Principal Components analysis calculates the variance-covariance matrix for a map list. New output bands are constructed in such a way that the largest variation is written to a new band 1, the second largest perpendicular variation to band 2, etc.

Factor Analysis
The Factor analysis operation calculates the correlation matrix for a map list. New output bands are constructed in such a way that the largest correlation is written to new band 1, the second largest perpendicular correlation in the second band, etc.

Variance-covariance matrix
The Variance-Covariance matrix operation calculates variances and covariances of raster maps in a map list. The variance is a means to express the variation of pixel values within a single raster map, i.e. a measure of the variation to the mean of the DN (Digital Number) values in a raster map. The covariance is a measure to express the variation of pixel values in two raster maps. It denotes the joint variation to the common mean of pixel values of the maps. Furthermore, the mean and standard deviation of each individual raster map is calculated.

Correlation matrix
The Correlation matrix operation calculates correlation coefficients of input raster maps of a map list. Correlation coefficients characterize the distribution of pixel values in two raster maps. Furthermore, the mean and standard deviation of each individual raster map is calculated.

Polygons

Neighbour polygons
The Neighbour polygons operation finds adjacent (or neighbouring) polygons in a polygon map and calculates the length of the boundaries of adjacent polygons.

Segments

Segment direction histogram
The Segment direction histogram operation calculates directions and lengths within segments, i.e. between all stored coordinate pairs of the segments. The output is a table with directions from 0 to 179° and the length and number of the segment parts in that direction. The results can be shown in a rose diagram.

Points

Spatial correlation
Spatial correlation calculates some point statistics: spatial autocorrelation (as Moran's I), spatial variance (as Geary's c) and semi-variance. Semi-variance is either calculculated in all directions (omnidirectional) or in a certain direction and the perpendicular direction (bidirectional). This may help you to get an impression of the nature of your point data, for instance prior to a point interpolation, and to find necessary input parameters for a Kriging operation.

As input for this operation, you can use a point map with a value domain, or a point map with a class or ID domain with an attribute table that contains one or more value columns (the attribute table must be linked to the map). The output of this operation is a table from which you can create graphs such as a semi-variogram.

Pattern Analysis
The Pattern analysis operation is a tool to obtain information on the spatial distribution of points in a point map. The output table contains six columns with the probabilities of finding 1 point (`Prob1Pnt`) within a certain distance from any point in your input map, then 2 points (`Prob2Pnt`), 3 points (`Prob3Pnt`), etc. Another column (`ProbAllPnt`) contains the sum of `Prob1Pnt`, `Prob2Pnt`,…, `Prob(n-1)`, in which `n` is the number of points in the input map.

By inspecting the graphs of distances against probabilities, you may recognize distribution patterns of your points like random, clustered, regular, paired etc.

INTERPOLATION

Raster

Densify raster map
The Densify raster map operation reduces the pixel size of your map. The number of rows and columns is increased and the new pixels in between the existing ones are assigned a value by means of a bilinear or bicubic interpolation.

You should use densify after a point interpolation. Further, densify can be used to improve the quality of printed raster maps.

Segments

Contour interpolation
Contour interpolation is an operation, which first rasterizes segments of a domain Value segment map, and then calculates values for pixels that are not covered by segments by means of a linear interpolation.

When using Contour interpolation on a segment map containing height (contour) information, the resulting raster map is a Digital Elevation Model.

Points

Point interpolation
In a point interpolation, the input map is a point map, and the output map is a raster map. The pixel values in the raster output map are interpolated from the point values.

There are four point interpolations, Nearest point, Moving average, Trend surface, and Moving surface:

Nearest point
The Nearest point operation requires a point map as input and returns a raster map as output. Each pixel in the output map is assigned the class name, identifier, or value of the nearest point, according to Euclidean distance. This method is also called Nearest Neighbour or Thiessen. The points in the input point map for the Nearest point operation do not need to be values necessarily; point maps (or attribute columns) with a class, ID or bool domain are also accepted.

For example, schools, hospitals, water wells, etc. can be represented by points. The output map of a nearest point operation on such a point map gives the 'service area' of the schools, hospitals or water wells, based on the shortest distance (as the crow flies) between points and pixels.
When you have many points or when you wish to use weights to indicate accessibilities, it is advised to rasterize the points, and then use Distance calculation to make a Thiessen map.

Moving average
The Moving average operation is a point interpolation, which requires a point map as input and returns a raster map as output. To the output pixels, weighted averaged point values are assigned.

The weight factors for the points are calculated by a user-specified weight function. The weight function ensures that points close to an output pixel obtain larger weights than points, which are farther away. Furthermore, the weight functions are implemented in such a way that points which are farther away from an output pixel than a user-defined limiting distance obtain weight zero.

When interpolating point values, it is for time efficiency reasons, strongly advised to choose a rather large pixel size for the output map. Further interpolation on the raster map values can be performed using the Densify operation or the Resample operation.

Trend surface
The Trend surface operation is a point interpolation which requires a point map as input and returns a raster map as output. One polynomial surface is calculated by a least squares fit so that the surface approaches all point values in the map. The calculated surface values are assigned to the output pixels.

Moving surface
The Moving surface operation is a point interpolation, which requires a point map as input and returns a raster map as output. For each output pixel, a polynomial surface is calculated by a least squares fit; for each output pixel, the surface will approach the weighted point values of the points which fall within the specified limiting distance.

Weight factors for the input points are calculated by a user-specified weight function. The weight function ensures that points close to an output pixel obtain larger weights than points, which are farther away. Furthermore, the weight functions are implemented in such a way that points which are farther away from an output pixel than a user-defined limiting distance obtain weight zero.

When interpolating point values, it is for time efficiency reasons, strongly advised to choose a rather large pixel size for the output map. Further interpolation on the raster map values can be performed using the Densify operation or the Resample operation.

Kriging
Kriging can be seen as a point interpolation, which requires a point map as input and returns a raster map with estimations and optionally an error map. The estimations are weighted averaged input point values, similar to the Moving Average operation. The weight factors in Kriging are determined by using a user-specified semi-variogram model (based on the output of the Spatial correlation operation), the distribution of input points, and are calculated in such a way that they minimize the estimation error in each output pixel. The estimated or predicted values are thus a linear combination of the input values and have a minimum estimation error. Two methods are available: Simple Kriging and Ordinary Kriging. The optional error map contains the standard errors of the estimates.

VECTOR OPERATIONS

Unique ID
The Unique ID operation can be used to assign a unique ID to all elements in a segment, polygon or point map. The result is an ID map that contains the same geographic information as the input map but now each point, segment or polygon has a unique ID. Further an attribute table is created which uses the same ID domain

as the output map; the table contains one column with the original classes, IDs or values of the input map. The domain of the table together with the column establishes the relation between the original classes, IDs or values in the input map and the output IDs.

Polygons

Attribute map of polygon map
By creating an attribute map of a polygon map, the class name or ID of each polygon in the original map is replaced by the value, class or ID found in a certain column in an attribute table.

A polygon map using a Class or ID domain, can have extra attribute information on the classes or identifiers in the map. These attributes are stored in columns in an attribute table. The attribute table can be linked to the map to which it refers, or to the domain of the map. You can check whether an attribute table is linked to the polygon map or to its domain through the Properties dialog box of the map or the domain.

Mask polygons
The Mask polygons operation allows you to selectively copy polygons of an input polygon map into a new output polygon map. The user has to specify a mask to select and retrieve the class names, IDs or values of the polygons that are to be copied.

Assign labels to polygons
The Assign labels to polygons operation can be used to recode polygons in a polygon map according to label points in a point map. For each label point, the surrounding polygon is determined; then the class name, ID, or value of the label point is assigned to that polygon.

Transform polygon map
The Transform polygon map operation transforms the XY-coordinate pairs of polygon boundaries in a polygon map from the map's current coordinate system to another target coordinate system. The Transform operation can only be used when a transformation between the coordinate systems is possible.

Segments

Attribute map of segment map
By creating an attribute map of a segment map, the class name or ID of each segment in the original map is replaced by the value, class or ID found in a certain column in an attribute table.

A segment map using a Class or ID domain can have extra attribute information on the classes or identifiers in the map. These attributes are stored in columns in an attribute table. The attribute table can be linked to the map to which it refers, or to the domain of the map. You can check whether an attribute table is linked to the segment map or to its domain through the Properties dialog box of the map or the domain.

Mask segments
The Mask segments operation allows you to selectively copy segments of an input segment map into a new output segment map. The user has to specify a mask to select and retrieve the class names, identifiers or values of the segments that are to be copied.

Assign labels to segments
The Assign labels to segments operation can be used to recode segments in a segment map according to label points in a point map. For each label point, the closest segment is determined; then the class name, ID or value of the label point is assigned to that segment.

Sub-map of segment map
The Sub-map of segment map operation copies a rectangular part of a segment map into a new segment map. The user has to specify minimum and maximum XY-coordinates for the new segment map.

Glue segment maps
The Glue segment maps operation glues or merges two or more segment maps into one output map. By default, the output map then comprises the total area of all input maps. The domains of the input maps are merged when needed.

For each input map, the user can specify a mask to select and retrieve the class names, IDs or values of the segments that are to be copied into the output map. The user can also specify a clip boundary, to copy only those segments to the output map which fall within the specified coordinate boundaries of the output map.

Densify segment coordinates
The Densify segment coordinates operation allows you to obtain more intermediate coordinates within segments. The segments of an input map are copied, and extra intermediate coordinates are added to the segments in the output map at a user-specified distance.
It is advised to use this operation before a Transform segments operation is performed.

Tunnel segments
The Tunnel segments operation reduces the amount of intermediate points within segments in a segment map. The segments of the input map are copied into a new segment map. However, for every three consecutive intermediate points within a segment, the middle one is omitted if it falls within a user-defined tunnel-width. Redundant nodes can also be removed.

Transform segment map
The Transform segment map operation transforms the XY-coordinate pairs of segments in a segment map from the map's current coordinate system to another

target coordinate system. The Transform operation can only be used when a transformation between the coordinate systems is possible.

Points

Attribute map of point map
By creating an attribute map of a point map, the class name or ID of each point in the original map is replaced by the value, class or ID found in a certain column in an attribute table.

A point map using a Class or ID domain can have extra attribute information on the classes or identifiers in the map. These attributes are stored in columns in an attribute table. The attribute table can be linked to the map to which it refers, or to the domain of the map. You can check whether an attribute table is linked to the point map or to its domain through the Properties dialog box of the map or the domain.

Mask points
The Mask points operation allows you to selectively copy points of an input point map into a new output point map. The user has to specify a mask to select and retrieve the class names, IDs or values of the points that are to be copied.

Sub-map of point map
The Sub-map of point map operation copies all points within a user-specified rectangle into a new point map. The user has to specify minimum and maximum XY-coordinates for the new point map.

Glue point maps
The Glue point maps operation glues or merges two or more point maps into one output map. By default, the output map then comprises the total area of all input maps. The domains of the input maps are merged when needed.

For each input map, the user can specify a mask to select and retrieve the class names, IDs or values of the points that are to be copied into the output map. The user can also specify a clip boundary, to copy only those points to the output map which fall within the specified coordinate boundaries of the output map.

Transform point map
The Transform point map operation transforms the XY-coordinate pairs of points in a point map from the map's current coordinate system to another target coordinate system. The Transform operation can only be used when a transformation between the coordinate systems is possible.

Transform coordinates
The Transform coordinates dialog box allows you to type in XY-coordinates or LatLon coordinates, using a certain input coordinate system; the operation will then show the resulting XY-coordinates or LatLon coordinates for another target coordinate system. The Transform coordinates dialog box can only be used when a transformation between the coordinate systems is possible.

RASTERIZE

Polygons to raster
The Polygons to raster operation rasterizes a polygon map. The class names, IDs, or values in the polygon map are also used in the raster map, i.e. the domain of the polygon map is also the domain of the raster map. The user has to select or create a georeference for the output raster map.

Segments to raster
The Segments to raster operation rasterizes a segment map. The class names, IDs, or values in the segment map are also used in the raster map, i.e. the domain of the segment map is also the domain of the raster map. The user has to select or create a georeference for the output raster map.

Segment density
The Segment density operation rasterizes a segment map. For each output pixel, the total length of segment parts within the boundaries the output pixel is summed: this is the output value for the pixel. By using a mask you can specify the elements of the input map that are to be used in the calculation.

Points to raster
The Points to raster operation rasterizes a point map. The class names, IDs, or values in the point map are also used in the raster map; i.e. the domain of the point map is also the domain of the raster map. The user has to select or create a georeference for the output raster map.

Point density
The Point density operation rasterizes a point map. For each output pixel, the number of points found in the pixel is counted: this is the output value for the pixel. This operation can be used to examine the regional distribution of points.

VECTORIZE

Raster to Polygons
The Raster to Polygons operation extracts polygons from units in a raster map. The output polygon map uses the same domain as the input raster map, i.e. the class names or IDs in the input raster map will also be used for the polygons in the output polygon map. No polygons are created for pixels with the undefined value.

Raster to Segments
The Raster to segments operation extracts segments from the boundaries of mapping units in a raster map. The segments in the output map either obtain the code Segments or a special code which is a combination of the class names or IDs of the two mapping units found on either side of the segment.

Raster to Points
The Raster to Points operation extracts a point from each pixel in the raster map. Each point gets the value, class name or ID of the corresponding pixel.

Polygons to Segments
The Polygons to Segments operation extracts polygon boundaries and creates a segment map out of them. A mask can be specified to extract segments of specific polygons.

Polygons to Points
The Polygon to Points operation creates a point for each polygon in the polygon map. Each point obtains the class name, ID, or value of the corresponding polygon. In this way, polygon label points are created. Optionally, you can also obtain label points for polygons, which have no class name, no ID or no value, i.e. for undefined polygons.

Segments to Polygons
The Segments to Polygons operation polygonizes a segment map. All segments in the segment map must be closed, i.e. connected to other segments or to themselves (islands) by nodes; dead ends are not allowed. A mask can be specified to polygonize specific segments.
Mind: to interactively polygonize segments, use the Polygonize option in the Segment editor.

Segments to Points
The Segments to Points operation creates a point map from a segment map. The output point map can contain either:
- a point for each node in the segment map, or
- points at a regular distance along the segments in the segment map, or
- points for all stored coordinate pairs in the segment map.

TABLE OPERATIONS

Transpose table
The Transpose table operation interchanges the rows and columns of a table. Each row of the input table becomes a column in the output table; while column names of the input table become output domain records.

Change domain of table
The Change domain of table operation copies the contents of an input table to a new table; the new table will have another domain than the input table.

For the domain of the output table, you can choose:
- domain `None`;
- an existing class or ID domain on disk;
- a class or ID domain of a column in the input table when that column contains unique classes or IDs;
- a class or ID domain of a column in the input table where the column does not contain unique classes or IDs and other column values need to be aggregated (average, minimum, maximum, sum, last value encountered).

Table to point map
The Table to point map operation creates a point map out of a table. The table should have at least two columns, which define the X- and Y-coordinates of the points.

You can choose between the following possibilities:
- the output point map should use the same domain as the table (ID domain); the table will be linked as attribute table to the output point map;
- the output point map will use the domain of a column in the table; the output point map will have no attribute table;
- the output point map should use a new ID domain which is based on the record numbers of the table (domain None) and a user-defined prefix; the table also obtains this new ID domain and the table will be linked as attribute table to the output point map. The new ID domain will automatically obtain the same name of the output point map.

Glue tables
The Glue tables operation allows you to glue or merge two or more tables together. As input tables, you may use:
- tables with domain `None`,
- tables with class domains,
- tables with ID domains,
- tables with class domains and ID domains.

The Glue tables operation should be regarded as a tool to combine different tables. You can for instance combine or integrate attribute tables of different years. Tables with domain `None` can also be glued vertically one below the other.

Appendix C

ILWIS expressions

Note: This appendix replaces Appendix E.2.1 in the ILWIS 2.1 Reference Guide.

Any ILWIS operation like Filter, Cross and Distance calculation, can be performed by typing an ILWIS expression on the command line of the Main window. You can also use these expressions in scripts.
In this topic, the syntax of expressions of operations is described.

For an overview of MapCalc and TabCalc expressions, see Appendices: ILWIS operators and functions (MapCalc/TabCalc).
For details on creating expressions on the command line and in scripts, see Appendices: construction of expressions.

For special script commands, see Appendices: ILWIS script language (syntax).

Introduction

The general syntax for expressions is:

```
 OUTMAP = expression
 OUTMAP := expression
```

The definition symbol (`=`) is used to create a dependent output object; the assignment symbol (`:=`) is used to create an editable object.

In the overview below:

- Bold	is used for expression names, the expression name is followed by parameters (between brackets, separated by commas);
- `Courier`	is used for obliged parts in expressions or in parameters;
- *Italics*	is used for parameters with special requirements, usually a short explanation follows;

- parameters:

'rasmap'	an input raster map;
'map list'	an input map list (set of raster maps with same domain and same georeference);
'pol map'	an input polygon map;
'segmap'	an input segment map;
'pntmap'	an input point map;
'table'	an input table;

'column'	an input column;
'domain'	an existing domain; the domain will be used for the output object;
'georef'	an existing georeference except georef `None`; the georeference will be used for the output raster map;
'coordsys'	any existing coordinate system;
'sample set'	an input sample set;
'newdomain'	the output domain that will be created by the expression;
'newgeoref'	the output georeference that will be created by the expression;

- a vertical bar | represents a choice;
- a paramater in square brackets [ ] represents an optional parameter;
- the phrase 'value map' or 'map with a value domain' means that the map should have domain of type Value.
- Any operation name in the list below starting with:

`Map`	creates an output raster map
`PolygonMap`	creates an output polygon map
`SegmentMap`	creates an output segment map
`PointMap`	creates an output point map
`Table`	creates an output table
`Matrix`	creates an output matrix

☞ Some operations need a value input map. When your raster map is of domain type Class, ID or Group, and an attribute table is linked to the map with one or more suitable value columns, then you may type 'map.column' on the command line instead of parameter 'map' listed below. The operation then directly uses the values of the attribute column.

List of ILWIS expressions

The list below follows the order of the Operations menu in the Main window. For more information about the individual operations, click the hyper-links of the operation names.

VISUALIZATION

Apply 3D
`MapApply3D`(*rasmap*, *georef3D*)

rasmap	input raster map cannot have georef `None`
georef3D	a georeference 3D

RASTER OPERATIONS

Map Calculate

expression	see MapCalc and TabCalc

Attribute Map

`MapAttribute`(*rasmap*, *column*) | *Rasmap.column*

rasmap	raster map with a Class, ID, or Group domain
column	column with Value, Class, ID, Group, Picture, or Color domain; by default a column from the attribute table of the raster map.

Cross

`MapCross`(rasmap1, *rasmap2*, output cross table)
`TableCross`(rasmap1, *rasmap2*)
`TableCross`(rasmap1, *rasmap2*, output cross rasmap)
`TableCross`(rasmap1, *rasmap2* [, output cross rasmap] [, `IgnoreUndefs` | `IgnoreUndef1` | `IgnoreUndef2`])

rasmap2 same georeference as input raster map1.

Aggregate Map

`MapAggregate`*AggFunc*(*rasmap*, *groupfactor*, *group* [,rowoffset, coloffset] [, newgeoref])
`MapAggregate`*AggFunc*(*rasmap*, *groupfactor*, *nogroup* [,rowoffset, coloffset])

AggFunc	`avg` | `cnt` | `max` | `med` | `min` | `prd` | `std` | `sum` When no aggregation function is specified, the upper left pixel of each block is used.
rasmap	raster map with a value domain for aggregate functions `avg`, `max`, `min`, `std`, `sum`; raster map with a class, ID, or value domain for aggregate function `med` ; raster map with any domain for aggregate function `prd`
groupfactor	a value (>= 1) to define the size of the blocks of input pixels to be aggregated; a value of 4 means that each block of 4 x 4 input pixels will be aggregated.
`group`	each block of input pixels is aggregated to 1 output pixel; a georeference factor is created with the same name as the output map.
`nogroup`	each block of input pixels is aggregated and the output value is stored in all pixels of the block. that correspond to the considered block of input pixels; the output map uses the same georeference as the input map.
rowoffset	optional parameter to skip the specified number of rows.
coloffset	optional parameter to skip the specified number of columns.
newgeoref	when using option `group`, optional parameter to specify the name of the output georeference; if not specified, the output georeference obtains the same name as the output map.

Distance

`MapDistance`(*source rasmap* [, *weight rasmap*| `1`])
`MapDistance`(*source rasmap*, [*weight rasmap*| `1`], output rasmap Thiessen)
`MapThiessen`(*source rasmap*[, *weight rasmap*| `1`], output rasmap Distance)

source rasmap	input raster map of any domain type; for all pixels with the undefined value, a distance value is calculated
[weight rasmap| 1]	weight map is an optional parameter to specify a map with weight factors; raster map of domain type Value. When a 1 is specified or when the parameter is not used, weight factor 1 is used for all pixels.
output rasmap Thiessen	name of output Thiessen raster map
output rasmap Distance	name of output Distance raster map

Iteration

`MapIter`(*StartMap*, *IterExpr* [, *nr of iterations*])

`MapIterProp`(*StartMap*, *IterExpr* [, *nr of iterations*])

StartMap	raster map that is used in the IterExpr.
IterExpr	an expression for neighbourhood operations.
nr of iterations	optional parameter to specify the maximum number of iterations; if not specified, the operation continues until no more changes occur.

Area Numbering

`MapAreaNumbering`(rasmap, `8|4` [, *newdomain*])

`8|4`	distinguish 8-connected or 4-connected areas; default is 8
newdomain	optional parameter to specify a name for the output ID domain; if not specified, the output domain will be stored by the output raster map (internal domain).

Sub Map

`MapSubMap`(rasmap, first row, first col [, nr rows, nr cols] [, *newgeoref*])

newgeoref	optional parameter to specify a name for the output georeference; if not specified, then the output georeference obtains the same name as the output map.

Glue Maps

`MapGlue`(*rasmap1*, *rasmap2* [, *rasmaps*] [, *newdomain*] [, `replace`])

rasmap1	input raster map which georeference will be used for the output raster map.
rasmap2	input raster map that will be resampled if needed.
rasmaps	optional parameter to specify more input raster maps. You can specify as many input raster maps as you like; comma delimited.
newdomain	optional parameter, when merging class or two ID maps, to specify the name of the output domain into which all items of the input domains are merged; if not specified, the output domain will be stored by the output raster map (internal domain).

`replace`	optional parameter to use for overlapping pixels the values of the last input map; if not specified, the values of the first input map are used for overlapping pixels.

Mirror Rotate

`MapMirrorRotate`(rasmap, *rotate type*)

rotate type	`mirrhor| mirrvert| mirrdiag|transpose| rotate90| rotate180| rotate270|normal`

IMAGE PROCESSING

Filter

`MapFilter`(*rasmap*, *filter*| *filter expression*)

rasmap	all filters use input raster maps with a value domain; the Majority and the UndefMajority filters also work on other domain types.
filter	`avg3x3` | `binmajor` | `conn8to4` | `d2fdx2` | `d2fdxdy` | `d2fdy2` | `dfddn` | `dfdup` | `dfdx` | `dfdy` | `dilate4` | `dilate8` | `edgesenh` | `inbnd4` | `inbnd8` | `laplace` | `lifegame` | `majority` | `majundef` | `majzero` | `med3x3` | `med5x5` | `outbnd4` | `outbnd8` | `peppsalt` | `shadow` | `shrink4` | `shrink8` | *user-defined filter on disk*
filter expression	`FilterLinear`(*rows,cols,expression*) | `Average`(*rows,cols*) | `RankOrder`(*rows,cols,rank*[,*threshold*]) | `Median`(*rows,cols*[,*threshold*]) | `Majority`(*rows,cols*) | `ZeroMajority`(*rows,cols*) | `UndefMajority`(*rows,cols*) | `Pattern`(*threshold*) | `FilterStandardDev`(*rows,cols*)
rows,cols	size of filter in rows and columns; value >= 1; maximum size of filter (rows×cols) <= 8000
threshold	if the difference between the resulting value and the original pixel value is smaller than or equal to the threshold, the calculated value is used. If the difference between the resulting value and the original pixel value is larger than the threshold, the original pixel value is retained.
rank	the rank number of which the pixel value is assigned to the central pixel.
expression	fill the values in the filter by an expression in which you can use the parameters `x`, `y`, and `r`.

Stretch

`MapStretch[Linear]`(*rasmap*, *range from*, *domain*)
`MapStretch[Linear]`(*rasmap*, *range from*, *domain*, *range to*)
`MapStretchHistEq`(*rasmap*, *range from*, *intervals*)

rasmap	input raster map using a value domain
range from	*min:max* | *perc*
perc	real value > 0
intervals	number of output intervals
domain	output value domain
range to	value range of output map as *min:max* | *min:max:prec*| ::*prec*

Slicing

`MapSlicing`(*rasmap*, domain group)

rasmap	input raster map using a value domain

Color Separation

`MapColorSep[aration]`(*rasmap*, *color*)| *rasmap.color*

rasmap	input raster map using a Picture domain or the Color domain
color	`red` | `green`| `blue` | `yellow` | `magenta` | `cyan` | `grey` | `gray` | `hue` | `saturation` | `sat` | `intensity` | `intens`

Color Composite

`MapHeckbert`(*map list*, *nr of colors*)
`MapColorComp[Linear]`(*map list*, *range1*, *range2*, *range3*)
`MapColorCompHistEq`(*map list*, *range1*, *range2*, *range3*)
`MapColorComp24[Linear]`(*map list* [, *range1*, *range2*, *range3*])
`MapColorComp24HistEq`(*map list*, *range1*, *range2*, *range3*)
`MapColorComp24HSI`(*map list*)

map list	existing map list which contains three raster maps of the Image domain or definition of map list as: `mlist(ImageRed, ImageGreen, ImageBlue)`
range1,2,3	min:max | perc
perc	0 ≤ real value < 50
nr of colors	2 ≤ integer value ≤ 255

Cluster

`MapCluster`(*map list*, *nr of clusters*)

map list	a map list which contains 1, 2, 3, or 4 raster maps that use the Image domain
nr of clusters	an integer value between 2 and 60 for the number of clusters in the output map.

Sample -

Classify

`MapClassify`(sample set, *classifier*)

classifier	`ClassifierBox`(factor) | `ClassifierMinDist`([threshold]) | `ClassifierMinMahaDist`([threshold]) | `ClassifierMaxLikelihood`([threshold])

Resample

`MapResample`(rasmap, georef, *resamp meth* `[`, `Patch` | `NoPatch` `]`)

resamp meth	`NearestNeighbour` | `BiLinear` | `BiCubic`

STATISTICS

Histogram

`TableHistogram`(*rasmap*)

rasmap	input raster map of any domain type. Mind: the output raster histogram table will always have the same name as the input raster map.

`TableHistogramPnt`(*pntmap*)

pntmap	input point map of any domain type. Mind: the output point histogram table will always have the same name as the input point map.

`TableHistogramPol`(*polmap*)

polmap	input polygon map of any domain type. Mind: the output polygon histogram table will always have the same name as the input polygon map.

`TableHistogramSeg`(*segmap*)

segmap	input segment map of any domain type. Mind: the output segment histogram table will always have the same name as the input segment map.

RASTER

Autocorrelation

`TableAutoCorrSemiVar`(rasmap, *max shift*)

max shift	maximum pixel shift; integer value > 0

MAP LIST

Principal Components

`MatrixPrincComp`(*map list*)

map list	map list containing raster maps with a value domain

Factor Analysis
`MatrixFactorAnal`(*map list*)

map list	map list containing raster maps with a value domain

POLYGONS

Neighbour Polygons
`TableNeighbourPol`(*polmap*)

polmap	input polygon map with a Class, ID or Group domain

SEGMENTS

Direction Histogram
`TableSegDir`(*segmap*)

POINTS

Spatial correlation
`TableSpatCorr`(*pntmap*)
`TableSpatCorr`(*pntmap*, *lagspacing* [, *direction* [, *tolerance* [, *bandwidth*]]])

pntmap	input point map with a value domain
lagspacing	parameter to specify length of linear distance intervals; if not specified, the output table will use logarithmic distance intervals
direction	optional parameter to find point pairs in this direction; clockwise angle from Y-axis; 0° ≤ direction ≤ 90°
tolerance	optional parameter to specify half of the opening angle with which points in the specified direction should be found; 0° < tolerance ≤ 45°
bandwidth	optional parameter to limit the tolerance angle to a certain maximum width

Pattern Analysis
`TablePattAnal`(*pntmap*)

pntmap	input point map with more than two points

INTERPOLATION

Densify map
`MapDensify`(*rasmap*, *factor*, *interpol meth*)

rasmap	raster map with a value domain for a BiLinear or Bicubic interpolation; raster map with any domain for Nearest Neighbour.
factor	real value > 1
interpol meth	`BiLinear` | `BiCubic` | `NearestNeighbour`

Contour Interpolation

`MapInterpolContour`(*segmap*, georef)

segmap	input segment map with a value domain

`MapInterpolContour`(*rasmap*)

rasmap	input raster map with a value domain; mind: algorithm only works well for rasterized contour lines

Point Interpolation

Nearest Point

`MapNearestPoint`(pntmap, georef)

Moving Average

`MapMovingAverage`(*pntmap*, georef, *weight func*)

pntmap	input point map with a value domain
weight func	`InvDist`(*Exp*,*LimDist*) | `Linear`(*Exp*,*LimDist*)
Exp	weight exponent
LimDist	limiting distance

Trend Surface

`MapTrendSurface`(*pntmap*, georef, *surface type*)

pntmap	input point map with a value domain
surface type	`Plane` | `Linear2` | `Parabolic2` | `2` | `3` | `4` | `5` | `6`

Moving Surface

`MapMovingSurface`(*pntmap*, georef, *surface type*, *weight func*)

pntmap	input point map with a value domain
surface type	`Plane` | `Linear2` | `Parabolic2` | `2` | `3` | `4` | `5` | `6`
weight func	`InvDist`(*Exp*,*LimDist*) | `Linear`(*Exp*,*LimDist*)
Exp	weight exponent
LimDist	limiting distance

Kriging

`MapKrigingSimple`(*pntmap*, georef, *semivarmodel*
[,errormap [, *remove duplic* [, *tolerance*]]])

`MapKrigingOrdinary`(*pntmap*, georef, *semivarmodel*, *limdist*
[,errormap [, *min*, *max* [, *remove duplic* [, *tolerance*]]]])

pntmap	input point map with a value domain
SemiVarModel	*Model(nugget, sill, range)* | `Power`*(nugget, slope, power)*
Model	`Spherical` | `Exponential` | `Gaussian` | `Wave` | `Circular` | `RatQuad`
limdist	limiting distance
min, max	optional parameter to specify the minimum and maximum number of points to be taken into account within the limiting distance
remove duplic	`no` | `average` | `firstval`, optional parameter to handle possible coinciding points

tolerance	optional parameter to specify a distance value in meters with which is determined whether points are coinciding or not

VECTOR OPERATIONS

Unique ID

`PolygonMapNumbering`(polmap [, *newdomain*])
`SegmentMapNumbering`(segmap [, *newdomain*])
`PointMapNumbering`(pntmap [, *newdomain*])

newdomain	optional parameter to specify a name for the output ID domain; if not specified, the output domain will be stored by the output map (internal domain).

POLYGONS

Attribute Map

`PolygonMapAttribute`(*polmap*, *column*)

polmap	polygon map with a Class, ID, or Group domain.
column	column with a Value, Class, ID, or Group domain; by default a column from the attribute table of the polygon map.

Mask Polygons

PolygonMapMask(polmap, *"mask"*)

"mask"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.

Assign Labels

`PolygonMapLabels`(polmap, pntmap)

Transform Polygons

`PolygonMapTransform`(polmap, coordsys)

SEGMENTS

Attribute Map

`SegmentMapAttribute`(*segmap*, *column*)

segmap	segment map with a Class, ID, or Group domain.
column	column with a Value, Class, ID, or Group domain; by default a column from the attribute table of the segment map.

Mask Segments

`SegmentMapMask`(segmap, *"mask"*)

"mask"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.

Assign Labels

`SegmentMapLabels`(segmap, pntmap [, *set domain*])

set domain	`yes` | `no` ; optional parameter to set the domain of the output segment map to the domain of the input segment map or to the domain of the input point map; if this parameter is not specified, the output segment map will use the same domain as the input segment map.
`no`	domain of output segment map is domain of input segment map.
`yes`	domain of output segment map is domain of input point map.

Sub Map

SegmentMapSubMap(segmap, minX, minY, maxX, maxY)

minX	minimum X-coordinate of output map
minY	minimum Y-coordinate of output map
maxX	maximum X-coordinate of output map
maxY	maximum Y-coordinate of output map

Glue Segment Maps

`SegmentMapGlue`(*segmap1*, *"mask1"*, *segmap2*, *"mask2"* [,...] [, *newdomain*])

`SegmentMapGlue`(*minX*, *minY*, *maxX*, *maxY*, *segmap1*,*"mask1"*, *segmap2*, *"mask2"* [, ...] [, *newdomain*])

segmap1,2, ...	are the input segment map names to be combined into one output segment map
"mask1"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.
minX	minimum X-coordinate of output map.
minY	minimum Y-coordinate of output map.
maxX	maximum X-coordinate of output map.
maxY	maximum Y- coordinate of output map.
newdomain	optional parameter, when merging Class or ID maps, to specify a name for the output domain into which all items of the input domains are merged; if not specified, the output domain will be stored by the output map (internal domain).

Densify Coords

`SegmentMapDensifyCoords`(segmap, *distance*)

distance	is the distance in meters at which extra intermediate points should be inserted into segments; real value > 0

Transform Segments

`SegmentMapTransform`(segmap, coordsys)

Tunneling

`SegmentMapTunneling`(segmap, *tunnel width*, *remove node*)

tunnel width	tunnel width in meters; real value >= 0
remove node	`yes` | `no` ; remove superfluous nodes or not

POINTS

Attribute Map

`PointMapAttribute`(*pntmap*, *column*)

pntmap	point map with a Class, ID, or Group domain
column	column with a Value, Class, ID, or Group domain; by default a column from the attribute table of the point map.

Mask Points

`PointMapMask`(pntmap, *"mask"*)

"mask"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.

Sub Map

`PointMapSubMap`(pntmap, minX, minY, maxX, maxY)

minX	minimum X-coordinate of output map
minY	minimum Y-coordinate of output map
maxX	maximum X-coordinate of output map
maxY	maximum Y-coordinate of output map

Glue Point Maps

`PointMapGlue`(*pntmap1*, *"mask1"*, *pntmap2*, *"mask2"* [,...] [, *newdomain*])

`PointMapGlue`(*minX*, *minY*, *maxX*, *maxY*, *pntmap1*, *"mask1"*, *pntmap2*, "mask2" [, ...] [, *newdomain*])

pntmap1,2, ...	are the input point maps to be combined into one output point map
"mask1"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.
minX	minimum X-coordinate of output map
minY	minimum Y-coordinate of output map
maxX	maximum X-coordinate of output map
maxY	maximum Y- coordinate of output map
newdomain	optional parameter, when merging Class or ID maps, to specify a name for the output domain into which all items of the input domains are merged; if not specified, the output domain will be stored by the output map (internal domain).

Transform Points

`PointMapTransform`(pntmap, coordsys)

RASTERIZE

Polygon to Raster

`MapRasterizePolygon`(polmap, georef)

Segment to Raster
`MapRasterizeSegment`(segmap, georef)

Segment Density
`MapSegmentDensity`(segmap, *"mask"*, georef)

"mask"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.

Point to Raster
`MapRasterizePoint`(pntmap, georef, *point size*)

point size	size in pixels; integer value > 0

Point Density
`MapRasterizePointCount`(pntmap, georef, *point size*)
`MapRasterizePointSum`(pntmap, georef, *point size*)

point size	size in pixels; integer value > 0

VECTORIZE

Raster to Polygon
`PolygonMapFromRas`(*rasmap* [, `8`|`4` [, `smooth`|`nosmooth`]])

rasmap	input raster map cannot have georef None
`8|4`	distinguish 8-connected or 4-connected areas; default is 8.
smooth	smooth polygon boundaries; default.
nosmooth	do not smooth polygon boundaries.

Raster to Segment
`SegmentMapFromRasAreaBnd`(*rasmap*, `8`|`4`, `smooth`|`nosmooth`,
`single`|`composite`)

rasmap	input raster map cannot have georef `None`
`8	4`	distinguish 8-connected or 4-connected areas
`smooth`	smooth segments
`nosmooth`	do not smooth segments
`single`	assign the name 'Segments' to all output segments (internal output domain).
`composite`	use the names of the pixels on both sides of the output segment and construct composite names for the output segment like `Agri|Forest` (internal output domain).

Raster to Point
`PointMapFromRas`(*rasmap*)

rasmap	input raster map cannot have georef None

Polygon to Segment

`SegmentMapPolBoundaries`(polmap, *"mask"*, `single`|`composite`)

"mask"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes.
`single`	assign the name 'Segments' to all output segments (internal output domain).
`composite`	use the names of the polygons on both sides of each output segment and construct composite names for the segments like `Agri`|`Forest` (internal output domain).

Polygon to Point

`PointMapPolLabels`(polmap [, `AlsoUndefs`])

`AlsoUndefs`	optional parameter to also obtain label points for polygons, which have no class name, ID or value.

Segment to Polygon

`PolygonMapFromSegment`(segmap [, *"mask"* [, `auto`]])

`PolygonMapFromSegment`(segmap [, *"mask"*, *domain* | *labelpntmap* [, `auto`]])

"mask"	a mask consists of (multiple) search strings; asterisks and question marks can be used as wild cards; on the command line, the total mask needs to be surrounded by double quotes; if not specified all segments are used.
domain	optional parameter to polygonize the segments to an existing domain; after polygonization, you can break the dependencies and edit the polygon map.
labelpntmap	optional parameter to polygonize the segments and use a point map with label points to assign names to the output polygons. If both the *domain* and *labelpntmap* parameters are not specified, the segments are polygonized and are assigned default names such as Pol 1, Pol 2, etc. (internal domain).
`auto`	optional parameter to automatically correct segments; deletes false polygons, deletes segments with dead ends, insert nodes when needed; if not specified and an error is found, the program stops and no polygon map is calculated.

Segment to Point

`PointMapSegCoords`(segmap)

`PointMapSegDist`(segmap, distance)

distance	distance interval in meters; real value > 0

`PointMapSegNodes`(segmap)

TABLE OPERATIONS

Transpose Table

`TableTranspose`(*table, col domain*)

`TableTranspose`(*table, col domain, valuerange*)

table	input table with a domain none, class or ID and maximum 1000 records
col domain	the domain that will be used for all columns in the transposed table
valuerange	if the column domain is a value domain, specify the value range as min:max : prec that will be used for all columns in the transposed table

Change Domain

`TableChangeDomain`(*table*, `None` | *domain*)

`TableChangeDomain`(*table*, *column*[, `avg` | `min` | `max` | `sum` | `last` | `no`])

domain	parameter to specify the name of an existing class or ID domain on disk. The output table will use this class or ID domain.
column	parameter to specify the name of a class or ID column in the input table; the output table will use the domain of the specified column. When no aggregation function is used, the classes or IDs in the specified column should be unique.
`avg` | `min` | `max` | `sum` | `last` | `no`	parameter to specify the aggregation method for other value columns, when the classes or IDs in the specified column are not unique: average, minimum value, maximum value, sum, last value encountered or no aggregation.

Table to PointMap

`PointMapFromTable`(table,[*colX*, *colY*,] coordsys [, *prefix* | *attribcol*])

colX, colY	parameters to specify names of columns in the table which contain the X- and Y-coordinates for the points; do not need to be specified when table contains columns with names X and Y.
prefix	optional parameter for a table with domain `None` to create a new ID domain for the output point map and table; if not specified, the new ID domain will use prefix `Pnt`.
attribcol	optional parameter to use a domain of a column in the table as the domain of the output point map.

Glue Tables

`TableGlue`(table1, table2 [, *more tables*] [, `vertical`])

`TableGlue`([*newdomain*,] table1, table2 [, *more tables*])

more tables	optional parameter to specify more input tables. You can specify as many input tables as you like; comma delimited.
`vertical`	in case the input tables have domain `None`: optional parameter to specify that tables should be merged vertically; if not specified, columns of the input tables will all appear as new columns in the output table (horizontally).

newdomain	in case the input tables have class or ID domains: optional parameter to specify a name for the domain of the output table; if not specified; the output domain will obtain the same name as the output table.

Appendix D

ILWIS expressions (alphabetic)

Note: This appendix replaces Appendix E.2.2 in the ILWIS 2.1 Reference Guide.

In the listing below, all ILWIS expressions are ordered by output object type, and then alphabetical.

Operations resulting in a raster map

`MapAggregate`*AggFunc*(rasmap, *groupfactor*, `group` [,*rowoffset*, *coloffset*] [,*newgeoref*])
`MapAggregate`*AggFunc*(rasmap, *groupfactor*, `nogroup` [,*rowoffset, coloffset*])
`MapApply3D`(rasmap, georef3D)
`MapAreaNumbering`(rasmap, `8|4` [, *newdomain*])
`MapAttribute`(rasmap, attribute column)
`MapCalculate`(expression)
`MapClassify`(sample set, *classifier*)
`MapCluster`(*map list, nr of clusters*)
`MapColorComp24[Linear]`(*map list* [*,range1, range2, range3*])
`MapColorComp24HistEq`(*map list, range1, range2, range3*)
`MapColorComp24HSI`(*map list*)
`MapColorComp[Linear]`(*map list, range1, range2, range3*)
`MapColorCompHistEq`(*map list, range1, range2, range3*)
`MapColorSep[aration]`(*rasmap,color*)
`MapCross`(rasmap1, rasmap2, output cross table)
`MapDensify`(rasmap, factor, interpolation method)
`MapDistance`(source rasmap [, weight rasmap`|1` [, output Thiessen rasmap]])
`MapFilter`(rasmap, *filter* | *filter expression*)
`MapGlue`(rasmap1, rasmap2 [, *rasmaps*][,*newdomain*] [, `replace`])
`MapHeckbert`(map list, *nr of colors*)
`MapInterpolContour`(segmap, georef)
`MapInterpolContour`(rasmap)
`MapIter[Prop]`(*StartMap*, IterExpr [, *nr of iterations*])
`MapKrigingOrdinary(`*pntmap*, georef, *semivarmodel, limdist* [, errormap [, m*in, max* [, *remove duplic* [, *tolerance*]]]])
`MapKrigingSimple(`*pntmap*, georef, *semivarmodel* [, errormap [, *remove duplic* [, *tolerance*]]])
`MapNearestPoint`(pntmap, georef)
`MapMirrorRotate`(rasmap, *rotate type*)

`MapMovingAverage`(pntmap, georef, *weight function*)
`MapMovingSurface`(pntmap, georef, *surface type, weight function*)
`MapRasterizePoint[Count | Sum]`(pntmap, georef, *pointsize*)
`MapRasterizePolygon`(polmap, georef)
`MapRasterizeSegment`(segmap, georef)
`MapResample`(rasmap, georef, resample method[, `Patch|NoPatch`])
`MapSegmentDensity`(segmap [, *"mask"*] , georef)
`MapSlicing`(rasmap, *domain* group)
`MapStretch[Linear]`(rasmap, *range from*, *domain* [, *range to*])
`MapStretchHistEq`(rasmap, *range from*, *intervals*)
`MapSubMap`(rasmap, first row, first col [, nr rows, nr cols] [, *newgeoref*])
`MapThiessen`(source rasmap [, weight rasmap|`1`], output Distance rasmap)
`MapTrendSurface`(pntmap, georef, *surface type*)

Operations resulting in a polygon map
`PolygonMapAttribute`(polmap, attribute column)
`PolygonMapFromRas`(rasmap [, `8|4` [, `smooth|nosmooth`]])
`PolygonMapFromSegment`(segmap, *"mask"* [, *domain*|*labelpntmap*] [, `auto`])
`PolygonMapLabels`(polmap, pntmap)
`PolygonMapMask`(polmap, *"mask"*)
`PolygonMapNumbering`(polmap [, *newdomain*])
`PolygonMapTransform`(polmap, coordsys)

Operations resulting in a segment map
`SegmentMapAttribute`(segmap, attribute column)
`SegmentMapDensifyCoords`(segmap, *distance*)
`SegmentMapFromRasAreaBnd`(rasmap`, 8|4, smooth|nosmooth, single|composite`)
`SegmentMapGlue`(segmap1, *"mask1"*, segmap2, *"mask2"* [, ...] [,*newdomain*])
`SegmentMapGlue`(*minX, minY, maxX, maxY*,segmap1, *"mask1"*, segmap2, *"mask2"* [, ...] [,*newdomain*])
`SegmentMapLabels`(segmap, pntmap, *set domain*)
`SegmentMapMask`(segmap, *"mask"*)
`SegmentMapNumbering`(segmap [, *newdomain*])
`SegmentMapPolBoundaries`(polmap, *"mask"*, `single|composite`)
`SegmentMapSubMap`(segmap, *minX, minY, maxX, maxY*)
`SegmentMapTransform`(segmap, coordsys)
`SegmentMapTunneling`(segmap, *tunnel width*, *remove nodes*)

Operations resulting in a point map
`PointMapAttribute`(pntmap, attribute column)
`PointMapFromRas`(rasmap)
`PointMapFromTable(`table, [*colX*, *colY*,] coordsys [,*prefix* | *attribcolumn*])
`PointMapGlue`(pntmap1, *"mask1"*, pntmap2, *"mask2"* [, ...] [,*newdomain*])

`PointMapGlue`(*minX, minY, maxX, maxY*,pntmap1, *"mask1"*, pntmap2, *"mask2"* [, ...] [,*newdomain*])
`PointMapMask`(pntmap, *"mask"*)
`PointMapNumbering`(pntmap [, *newdomain*])
`PointMapPolLabels`(polmap | `AlsoUndefs`)
`PointMapSegCoords`(segmap)
`PointMapSegDist`(segmap, *distance*)
`PointMapSegNodes`(segmap)
`PointMapSubMap`(pntmap, *minX, minY, maxX, maxY*)
`PointMapTransform`(pntmap, coordsys)

Operations resulting in a table
`TableAutoCorrSemiVar`(rasmap, *pixel shift*)
`TableChangeDomain`(table, `None` | *domainondisk*)
`TableChangeDomain`(table, columnname [, `avg`| `min`| `max`| `sum`| `last`| `no`])
`TableCross`(rasmap1, rasmap2)
`TableCross`(rasmap1, rasmap2, output cross rasmap)
`TableCross`(rasmap1, rasmap2[, output cross rasmap] [,`IgnoreUndefs` | `IgnoreUndef1` | `IgnoreUndef2`])
`TableGlue`(table1, table2 [, *more tables*] [, `vertical`])
`TableGlue`([*newdomain*,] table1, table2 [, *more tables*])
`TableHistogram`(rasmap)
`TableHistogramPnt`(pntmap)
`TableHistogramPol`(polmap)
`TableHistogramSeg`(segmap)
`TableNeighbourPol`(polmap)
`TablePattAnal`(pntmap)
`TableSegDir`(segmap)
`TableSpatCorr`(pntmap[, *lagspacing* [, *direction* [, *tolerance* [, *bandwidth*]]]])
`TableTranspose`(table, column *domain* [,*value range*])

Operations resulting in a matrix
`MatrixFactorAnal`(map list)
`MatrixPrincComp`(map list)

Aggregation or join operations in tables resulting in a column
`ColumnAggregateAvg`(column [, *group* [, *weight*]])
`ColumnAggregateCnt`(column [, *group*])
`ColumnAggregateMax`(column [, *group*])
`ColumnAggregateMed`(column [, *group* [, *weight*]])
`ColumnAggregateMin`(column [, *group*])
`ColumnAggregatePrd`(column [, *group* [, *weight*]])
`ColumnAggregateStd`(column [, *group* [, *weight*]])
`ColumnAggregateSum`(column [, *group*])

`ColumnJoin`(table, column)
`ColumnJoin`(table, column, *key1*)
`ColumnJoin2ndKey`(table, column, *viakey2*)
`ColumnJoin2ndKey`(table, column, *key1*, *viakey2*)
`ColumnJoin`*`AggFunc`*(table, column, *groupbykey2*)
`ColumnJoin`*`AggFunc`*(table, column, *groupbykey2*, *weight*)
`ColumnJoin`*`AggFunc`*(table, column, *groupbykey2*, *weight*, *key1*)

Special syntax to create attribute maps
Map.column
Polygonmap.mpa.column
Segmentmap.mps.column
Pointmap.mpp.column

Special syntax to perform color separation
Map.color

Appendix E

ILWIS script language (syntax)

Note: This appendix replaces Appendix F in the ILWIS 2.1 Reference Guide.

A script is a sequenced list of ILWIS expressions. By creating a script, you can build a complete GIS or Remote Sensing analysis for your own research discipline. Scripts are more or less equivalent to batch files in ILWIS version 1.4.

General information

In a script, you can use any MapCalc or TabCalc expression, any expression for an operation, you can use parameters, you can call other scripts, you can display ILWIS objects on the screen, and further you can use a number of commands for file management, to handle object properties, to break dependencies and release disk space, to edit class or ID domains, etc.

When you run a script, no dialog boxes appear and no questions are asked; all lines in the script are simply performed. Error messages appear in case syntax errors are detected in a MapCalc expression, in a TabCalc expression, or in an expression for another operation, or in a script command. Further error messages appear when a script command is not recognized, or when required objects are not found. A script line is ignored when the syntax is correct and necessary objects are found but the command cannot be performed otherwise (e.g. creating objects that already exist, missing or wrong extensions during a copy).

Parameters in scripts

Parameters in a script can replace (parts of) object names, operations, etc. Parameters in scripts work as DOS replaceable parameters in DOS batch files, and must be written in the script as `%1, %2, %3,` up to `%9`. The parameters of a script have to be filled out on the command line when you run the script. The first text string found after the script name will replace `%1` in the script, etc. For more information, see How to use parameters in scripts.

To run a script

To run a script from the command line of the Main window, type:
Run scriptname parameter parameter.
If a script has no parameters, you can directly double-click the script in the Catalog. For more information, see How to run scripts.

Example
An example of a script is presented in Map and Table calculation : creating and running scripts.

☞ Single text lines of a script, i.e. the commands and expressions described below, can also be typed on the command line of the Main window. To avoid any dialog boxes, it is advised to use a semicolon (';') at the end of such a line. In a script, semicolons are not allowed.

Expressions for calculations and other operations

Most text lines in a script will consist of MapCalc and TabCalc expressions and expressions of operations that you can also type on the command line of the Main window or on the command line of a table window. You should be familiar with these expressions. For an overview of MapCalc and TabCalc operators and functions, refer to Appendices: MapCalc and TabCalc operators and functions. For an overview of expressions for other operations, refer to Appendices: ILWIS expressions. For more information on the creating of expressions, see Appendices: construction of expressions.

MapCalc and Tabcalc

For MapCalc expressions, no special syntax is required: you can simply type the MapCalc expression as you would type it on the command line of the Main window. For example, to `sum` maps `map1` and `map2` to create `map3`, type in the script:

```
map3=map1+map2
```

For TabCalc expressions, it is necessary that you type `tabcalc` ***`tablename`*** in front of the tabcalc expression. For example, to `sum` columns `col1` and `col2` in table `MyTable` and to store the results in column `col3`, type in the script:

```
tabcalc MyTable col3=col1+col2
```

If you like, you can also perform table calculations on other objects that can be opened as a table, e.g. histograms, point maps, class representations. Then, specify the extension of the object after the table name:

```
tabcalc tablename.ext a=b+c
```

If you want to perform a series of table calculations in one table, it is advised to use the following script commands:

```
opentbl tablename.ext
```

Keep the table *tablename.ext* open as the first line before a series of TabCalc expressions on one table.

```
closetbl tablename.ext
```

Close the open table *tablename.ext* as the last line after a series of TabCalc expressions on one table.

ILWIS operations

To perform other ILWIS operations, you can use any ILWIS expression as described in Appendices C: ILWIS expressions and D: ILWIS expressions (alphabetic).

E.1 Additional script commands

A number of additional script commands is available for file management, to show objects, handle object properties, edit object properties, create objects, calling other scripts, etc.

In many of the following script commands, object names and extensions of their object definition files must be specified. In some script commands, you are allowed to use wildcards `*` and `?` to specify object names and their extensions (*object.ext* and *table.ext.col*).

When using a script command that works on a column in a table (*table.ext.col*), you can ignore the extension when the table has extension .TBT. Table extensions only need to be specified when the column is stored in a histogram, a point map, a class representation, a georeference tiepoints, etc.

Further, in the list below, optional parameters of script commands are shown between square brackets. Omit these square brackets when writing a script. Square brackets are only recognized for TabCalc expressions to indicate a specific record in a table.

Remarks, comments, messages and pause

`rem` *This is a remark*	All text on this line after `rem` is ignored by the script. In this manner, you can document your script expressions.
`//` *This is a remark*	All text on this line after `//` is ignored by the script. In this manner, you can document your script expressions.

`begincomment`
line 1 of my multiple line comment
line 2 of my multiple line comment
`endcomment`
	All lines of text between the commands `begincomment` and `endcomment` are ignored by the script. In this manner, you can document your script expressions and you can temporarily exclude parts of your script.

`message` *Message on my screen*	Obtain a message box on the screen with any text; the text can be as long as you like. After pressing the OK button in the message box, the script will continue. In this manner, you can display texts on the screen during demos, etc.
`pause` *seconds*	Stop the script for a certain amount of time (seconds). You can use this command for instance when you want to show multiple maps and give the user time to view each one of them.

Open/Show an object

`show` *object.ext*	Open/show object *object.ext.*
`open` *object.ext*	Open/show object *object.ext.*
`open –noask` *object.ext*	Open/show object *object.ext* with its default display options, i.e. a Display Options dialog box will be skipped. You can use this command for instance to quickly show a map on the screen.
`closeall`	Close all ILWIS windows except the ILWIS Main window. You can use this command for instance to close any map and/or table windows being displayed on the screen.

File management

`cd` *path*	Change directory to directory *path.*
`cd` *d:path*	Change drive to *drive d*: and change directory to directory *path.*
`md` [*drive*:]*path*	Make directory *path*. Optionally make directory *path* on drive *drive*
`mkdir` [*drive*:]*path*	Make directory *path*. Optionally make directory on *path* on drive *drive.*
`rd` [*drive*:]*path*	Remove directory *path*. Optionally remove directory *path* from drive *drive.*
`rmdir` [*drive*:]*path*	Remove directory *path*. Optionally remove directory *path* from drive *drive.*
`rd` [*drive*:]*path* `-force`	Remove directory *path* while deleting all files in that directory. Optionally remove directory *path* and all files in that directory from drive *drive.*
`rmdir` [*drive*:]*path* `-force`	Remove directory *path* while deleting all files in that directory. Optionally remove directory *path* and all files in that directory from drive *drive.*

An error message appears when changing to a directory that does not exist, or when removing a directory that does not exist. The script line is ignored, when making a directory that already exists, or when deleting files that do not exist.

`copy` *object.ext objname*	Copy object *object.ext* to new name *objname*.
`copy` *object.ext objname* `-breakdep`	Copy object *object.ext* to new name *objname* while breaking the dependency links of *object.ext*.
`copy` *object.ext path*	Copy object *object.ext* to existing directory *path*. Wildcards are allowed.
`copy` *object.ext path* `-breakdep`	Copy object *object.ext* to existing directory

	path while *breaking the dependencies of object.ext*. Wildcards are allowed.
`copyfile` *file.ext filename.ext*	Copy file *file.ext* to new file *filename.ext*.
`copyfile` *file.ext path*	Copy file *file.ext* to existing directory *path*. Wildcards are allowed.

When copying objects (or files), you cannot copy objects to another directory and give the object another name at the same time.
When copying an object to another directory, existing objects in that directory are not overwritten. In the same way, when copying an object in the current directory to an object name, which already exists, the script line is ignored.

`del` *object.ext*	Delete object *object.ext*. Wildcards are allowed.
`del` *object.ext* `-force`	Tries to delete object *object.ext* which is not completely valid (i.e. an error occurs when the object is opened). Wildcards are allowed.
`delcol` *table.ext.column*	Delete column *table.ext.column*.
`delfile` *file.ext*	Delete file *file.ext* as if this file was deleted in DOS or the File Manager. Wildcards are allowed.

The del and delcol commands check whether the object is not read-only or whether a column is not table-owned; these commands do not take into account whether an object is still used by other objects. The script line is ignored when objects, columns or files do not exist.

Handling properties of dependent objects

`update` *object.ext*	Make the dependent map or table *object.ext* up-to-date. Wildcards are allowed.
`updatecol` *table.ext.column*	Make dependent column *table.ext.column* up-to-date.
`breakdep` *object.ext*	Break the dependency links of dependent map or table *object.ext*. Wildcards are allowed.
`breakdep` *object.ext* `-force`	Tries to break the dependency links of dependent map *object.ext* which is not completely valid (i.e. an error occurs when the object is opened). Wildcards are allowed.
`breakdepcol` *table.ext.column*	Break the dependency links of dependent column *column* in table *table.ext*.
`reldisksp` *object.ext*	Delete the data file(s) of dependent object *object.ext*.
`calc` *object.ext*	Recalculate the data files of dependent map or table *object.ext*. Wildcards are allowed.
`calccol` *table.ext.col*	Recalculate dependent column *table.ext.col*.

Editing properties of editable source objects (advanced)

`changedom` *object.ext domname* [*valuerange*]	Change the domain of raster, polygon, segment or point map *object.ext* to existing domain *domname*, while converting the class names, IDs, or values of the original domain into new domain *domname*. Optionally, in case of a value domain *domname*, set the value range of the object to valuerange. Specify the value range as *min:max:precision*, as *min:max*, or as *::precision*. This is a rather safe way to change the domain of a map into another domain or to change the precision of a map.
`setdom` object.ext *domname* [*valuerange*] [`-force`]	Set the domain of *object.ext* to *domname*. Wildcards are allowed for *object.ext*. Optionally, in case of a value domain *domname*, set the value range of the object to *valuerange*. Specify the value range as *min:max:precision*, as *min:max*, or as *::precision*. When you also specify the `-force` flag, no checks are performed.
`setvr` *object.ext valuerange*	Set the value range of *object.ext* to *valuerange*. Wildcards are allowed for *object.ext*. Specify the value range as *min:max:precision*, as *min:max*, or as *::precision*.
`setgrf` *rasmap georef*	Set the georeference of raster map *rasmap* to *georef*. Wildcards are allowed for *rasmap*.
`setcsy` *map.ext coordsys*	Set the coordinate system of point, segment or polygon map *map.ext* to *coordsys*. Wildcards are allowed for *map.ext*.
`setcsy` *georef coordsys*	Set the coordinate system of georeference *georef* to *coordsys*. Wildcards are allowed for *georef*.
`setreadonly` *object.ext*	Mark object *object.ext* as read only. Read only objects cannot be edited or deleted. Wildcards are allowed.
`setreadwrite` *object.ext*	Remove the read only flag for object *object.ext*: the objects are editable and deleteable. Wildcards are allowed.
`setatttable` *map.ext atttable*	Set the attribute table of class or ID map *map.ext* to *atttable*. Wildcards are allowed for *map.ext*.

`setatttable` *map.ext*	Remove the link between class or ID map *map.ext* and its attribute table. Wildcards are allowed.

The `setdom`, `setvr`, `setgrf`, `setscy` and `setatttable` commands are only performed on objects that are not read only.

Creating (empty) maps and tables

`crmap` *map georef domname*	Create raster map map with existing georeference georef and existing domain domname.
`crsegmap` *map crdsys domname*	Create segment map with existing coordinate system crdsys and existing domain domname.
`crpntmap` *map crdsys domname*	Create point map map with existing coordinate system crdsys and existing domain domname.
`crtbl` *table domname*	Create table table using existing Class or ID domain domname.
`crtbl` *table nrrecs*	Create table table using domain None and with a number of records specified as nrrecs.

Creating domains

`crdom` *domname* [`-type=class|ID|group`] `-items`=*number* [`-prefix`=prefix]

Create domain *domname* with a number of items specified as `-items`=*number*. Optionally, specify `-type=class` to create a class domain, or `–type=ID` to create an ID domain, or `–type=Group` to create a group domain. If parameter `-type` is omitted, then a class domain is created. Optionally specify `-prefix`=*prefix* to obtain classes or IDs with a certain *prefix*. If parameter `-prefix` is omitted when creating a class domain, classes will obtain prefix `class`, thus `class 1`, `class 2`, etc. If parameter `-prefix` is omitted when creating an ID domain, the IDs will obtain prefix `ID`, thus `ID 1`, `ID 2`, etc.

`crdom` *domname* `-type=value` `-min`=*value* -max=*value* [`-prec`=*value*]

Create value domain *domname* with a specified value range between `min`=*number* and `max`=*number*. Optionally, specify a precision for the value domain as `-prec`=value.

Examples:

`crdom` *domname* `-type=class -items=10 -prefix=cl`
: Create class domain *domname* and add ten items to this domain with class names "cl 1", "cl 2"…"cl 10".

`crdom` *domname* `-type=id –items=100 -prefix=prov`
: Create ID domain *domname* and add hundred items to this domain with IDs "prov 1", "prov 2"…"prov 100".

`crdom` *domname* `-items=0`
: Create class domain *domname* without any items. You can add items with the `additemtodomain` command.

`crdom` *domname* `-type=value -min=100 -max=200`
: Create value domain *domname* with a value range between 100 and 200 (precision is 1).

`crdom` *domname* `-type=value -min=10 -max=20 -prec=0.01`
: Create value domain *domname* with a value range between 10.00 and 20.00 and a precision of 0.01.

The `crdom` command is ignored when domain domname already exists.

Editing a class or ID domain

`additemtodomain` *domname class* [*classcode*]
: Add item class, optionally with code *classcode*, to class or ID domain *domname*. Class names, which contain spaces, must be enclosed by double quotes.

`additemtodomaingroup` *domname upperlimit class [classcode]*
: Add an item to group domain *domname*. The item is defined by an *upperlimit* (a real value), a class name *class* and, optionally, a *classcode*

`mergedom` *domname1 domname2*
: Merge the items of class or ID domain *domname2* into class or ID domain domname1.

Create representations

`crrpr` *rprname domname*
: Create representation *rprname* for class or value domain *domname*.

The `crrpr` command is ignored when representation *rprname* already exists.

Create georeference corners

```
crgrf grfname nrrows nrcols [-crdsys=coordsysname]
        -lowleft=(minX,minY)
        -upright=(maxX,maxY)|-pixsize=value
        [-centercorners+]
```

Create georeference corners *grfname* with the number of rows as specified by *nrrows*, the number of columns as specified by *nrcols*, the coordinates for the lower left corner as specified by `-lowleft`=(*minX*, *minY*), and either the coordinates for the upper right corner specified by `-upright`=(*maxX*, *maxY*) or the pixel size as specified by `-pixsize`=value.
Mind: in a script, you are not allowed to use spaces within a coordinate expression (X,Y). Optionally, a coordinate system *coordsysname* can be specified by using parameter `–crdsys` =*coordsysname*, otherwise the system coordinate system `Unknown` will be used. Furthermore, you can optionally add coordinates to the centers of the corner pixels by using parameter `-centercorners+`. When this parameter is not used, coordinates will be added to the corners of the corner pixels.

Examples:

`crgrf` *grfname* `500 1000 -crdsys=cs -lowleft=(0,0)`
`      –upright =(5000,10000)`

Create georeference corners *grfname* with 500 rows and 1000 columns and using coordinate system `cs`. The coordinate boundaries are defined by the lower left coordinate (0,0) and the upper right coordinate (5000,10000).

`crgrf` *grfname* `500 1000 -crdsys=cs -lowleft=(0,0)`
`      –pixsize=10`

Create georeference corners *grfname* with 500 rows and 1000 columns and using coordinate system `cs`. The georeference has as lower left coordinate (0,0) and as pixel size 10 m.

Mind: In a script, you are NOT allowed to use spaces within a coordinate expression (X,Y). The `crgrf` command is ignored when georeference *grfname* already exists.

Creating a two-dimensional table

`cr2dim` *2dimtablename indomname1 indomname2 outdomname3 [valuerange]*

Create two-dimensional table *2dimtablename* using existing input domains *indomname1* and *indomname2* and use existing domain *outdomname3* as the domain of the fields in the table. If *outdomname3* is a value domain, you can optionally specify the value range of this domain as *min:max* or as *min:max:precision*.

Converting domains

`domclasstoid` *domname[.ext]*
Convert class domain *domname* into an ID domain.

`domidtoclass` *domname[.ext]*
Convert ID domain *domname* into a class domain.

`dompictoclass` *domname[.ext]*
Convert Picture domain *domname.ext* into a class domain.

These are rather safe ways to convert one domain into another. When the domain you want to convert is an internal domain, which is stored in a map, you need to specify the extension of the map after the domain name.

Calling other scripts

`run` *script2*	Run another script named *script2* (without parameters).
`run` *script2 param1 param2*	Run another script name *script2*; fill out parameters.

If *script2* is not found, an error message appears.

Start other Windows applications

`!`*Command line*	Performs *Command line* as if entered in the Windows (File) Run dialog box. Starts any Windows application, batch file, or DOS application (with a .PIF file available). Applications should have one of the following extensions: `.exe`, `.com`, `.bat`, `.pif`. Type the application name directly after the exclamation mark (no spaces allowed). Example: to start Word and open document MyDoc, type: `!Winword MyDoc`

Import files from ILWIS 1.4 to ILWIS 2.

`Import14` *file14.ext* [*outputdir*]
Batch-wise import of *file14.ext* in the current directory, or optionally to the specified output directory *outputdir*. Wildcards are allowed. The ILWIS 2 object(s) keep the name(s) of the 1.4 file(s); new extensions are created

during import. Domains, representations, georeferences etc. are created using defaults.

`Import14` *file14.ext* [*ilwis2name|outputdir*] [`-dmt`=*domtype*] [`-dom`=*domainname*] [`-grf`=*georef*] [`-att`=*tablename*]

Import an ILWIS 1.4 file *file14.ext* according to your wishes. All parameters shown above between square brackets can but do not have to be used.

ilwis2name|outputdir	Either specify an ILWIS 2 object name as *ilwis2name* for the 1.4 file to be imported or specify an output directory as *outputdir* in which the imported object should appear. If this parameter is omitted, then *file14.ext* is imported in the current directory and the ILWIS 2 object(s) keep the name(s) of the 1.4 file(s).
`-dmt`=*domtype*	Specify the domain type *domtype* for the output object as: `Picture`, `Image`, `Value`, `Class`, or `ID`. If you specify domain type Class or ID, you also have to use the `-dom` option to specify the name of that domain.
`-dom`=*domname*	If you specified domain type `Class` or `ID` in the previous option, then also specify a new or existing *domname* for the output object.
`-grf`=*georef*	When importing a raster map, you can specify the name of a new or existing georeference *georef*.
`-att`=*tablename*	When importing a 1.4 point table which contains besides the `X!`, `Y!` and `Name$` columns also attribute columns, you can specify the name of the ILWIS 2 attribute table as *tablename*. The point table is then imported as an ILWIS 2 map and the other columns (i.e. columns other than `X!, Y!` and `Name$`) are imported as an attribute table of the point map.

Example:

```
Import14 orotm4.mpd -dmt=image -grf=tmgeoref
```

Import 1.4 raster map `Orotm4` as an image using existing georeference tiepoints `tmgeoref`.

Importing files into ILWIS

`import` format(file.ext, ilwobj)	Import a data file *file.ext* into an ILWIS object with the name *ilwobj*. The extension of the input file *file.ext* **must** be specified. The extension for the output ILWIS object(s) *ilwobj*

	will be automatically created during import. For *format*, you have to specify one of the following formats: `arcinfonas` | `ascii` | `bmp` | `bna` | `dbase` | `dxf` | `e00` | `erdas` | `gif` | `ida` | `idrisi` | `ilwis14pnt` | `ilwis14pol` | `ilwis14ras` | `ilwis14seg` | `ilwis14tbl` | `infocam` | `lin` | `pcx` | `shape` | `smt` | `tiff`.
`arcinfonas`	Import an Arc/Info non-compressed ASCII raster file to an ILWIS raster map.
`ascii`	Import an ILWIS 1.x ASCII raster file (.ASC) to an ILWIS raster map.
`bmp`	Import a Windows bitmap (.BMP) to an ILWIS raster map.
`bna`	Import an Atlas vector data file (.BNA) to an ILWIS segment map.
`dbase`	Import a dBase III/IV file (.DBF) to an ILWIS table. This option will come up with the Import dBase III/IV table dialog box.
`dxf`	Import an AutoCad .DXF file to an ILWIS point and/or segment and/or polygon map.
`e00`	Import an Arc/Info file in interchange format (.E00) to an ILWIS raster and/or polygon and/or segment and/or point map. When attributes are available, also an ILWIS table will be created.
`erdas`	Import an Erdas .GIS file into an ILWIS raster map; or import an Erdas .LAN file into a single ILWIS raster map or into an ILWIS map list containing multiple raster maps (bands).
`gif`	Import a gif image (.GIF) to an ILWIS raster map.
`ida`	Import an IDA image (.IMG) to an ILWIS raster map.
`idrisi`	Import an Idrisi image (.DOC). to an ILWIS raster map.
`ilwis14pnt`	Import an ILWIS 1.x point table (.PNT) to an ILWIS 2 point map.
`ilwis14pol`	Import an ILWIS 1.x polygon map (.POL) to an ILWIS 2 polygon map.
`ilwis14ras`	Import an ILWIS 1.x raster file (.MPD) to an ILWIS 2 raster map.
`ilwis14seg`	Import an ILWIS 1.x segment maps (.SEG) to an ILWIS 2 segment map.
`ilwis14tbl`	Import an ILWIS 1.x table (.TBL) to an ILWIS 2 table.

`infocam`	Import an Infocam sequential file (.SEQ) to an ILWIS point and/or segment and/or polygon map. When attributes are available, also an ILWIS table will be created.
`lin`	Import an Arc/Info file created with the Ungenerate command (.LIN and .PTS) to an ILWIS segment and/or point map. When the extension of the Arc/Info file is .PTS, you will obtain a point map, otherwise you will obtain a segment map.
`pcx`	Import a PaintBrush image (.PCX) to an ILWIS raster map.
`shape`	Import Arc/View Shape files (.SHP, .SHX, DBF) to an ILWIS point and/or segment and/or polygon map. Furthermore, an ILWIS table will be created.
`smt`	Import an ILWIS 1.x ASCII vector file (.SMT) to an ILWIS segment map.
`tiff`	Import a Tiff image (.TIF) to an ILWIS raster map. When the Tiff image contains GeoTiff information, a georeference will be created for the imported map; furthermore, it is attempted to create a coordinate system (you can find projection information in the description of the coordinate system).

Examples:

`import` erdas(soil.gis, ilwsoil)

Imports the Erdas file SOIL.GIS into ILWIS; a raster map with the name ILWSOIL.MPR will be created. The import will also look for the availability of an Erdas file called SOIL.TRL that may accompany the .GIS file.

`import` e00(parcel.e00, ilwparc)

Imports the Arc/Info interchange file PARCEL.E00 into ILWIS; a segment map, a polygon map, a raster map and a point map can be created (all named ILWPARC), where the vector maps can also be linked to attribute tables.

For more information on Import, see also the Import dialog box, or the `import.def` file in your ILWIS directory.

☞ Besides using the Import command in a script, you can also use it within ILWIS on the command line; then, the complete command must be followed by a semicolon.

Export files from ILWIS 2 to ILWIS 1.4

`export14` *object2.ext name14*

Export ILWIS 2 raster map, polygon map, segment map, point map or table *object2.ext* to an ILWIS 1.4 file *name14*.

Exporting ILWIS 2 maps and tables

`export` *format*(*ilwobj.ext*, *filename*)

Export an ILWIS 2 map or table *ilwobj.ext* to another data file *filename*.The extension of the ILWIS 2 map or table *ilwobj.ext* **must** be specified; `.mpl` for a map list, `.mpr` for a raster map, `.mpa` for a polygon map, `.mps` for a segment map, `.mpp` for a point map and `.tbt` for a table. Furthermore, ILWIS 2 objects which are stored as a table can be exported as tables: histograms (`.his`, `.hsa`, `.hss`, `.hsp`), 2-dimensional tables (`.ta2`), domain class/ID/group (`.dom`), representation class (`.rpr`), georeference tiepoints (`.grf`), coordinate system tiepoints (`.csy`). The extension(s) for the output data file(s) filename will be automatically created during export. For format, you have to specify one of the following formats: `arcinfonas` | `arcinfopts` | `ascii` | `bmp` | `bna` | `cartcv` | `cartvte` | `dbase` | `dbasesdf` | `delimited` | `dxf` | `e00` | `erdasdig` | `erdasgis` | `erdaslan` | `geosoft` | `gif` | `gina` | `hpgl` | `ida` | `idrisi` | `ilwis14` | `infocam` | `lin` | `pcx` | `shapefile` | `sif` | `smt` | `themak2` | `tiff` | `usemap`.

`arcinfonas`	Export an ILWIS raster map to an Arc/Info non-compressed ASCII file (.NAS). Internally, the Convert14 program is used.
`arcinfopts`	Export an ILWIS point map to an Arc/Info .PTS (ASCII) file. From this file, (label) points can be created using the Arc/Info Generate command. Instead of the command `arcinfopts`, you can also use the command `arcgen`.
`ascii`	Export an ILWIS raster map to an ILWIS 1.x ASCII file.
`bmp`	Export an ILWIS raster map to a Windows bitmap (.BMP).

`bna`	Export an ILWIS segment map to an Atlas vector map (.BNA).
`cartcv`	Export an ILWIS polygon or segment map to a Cart/o/graphix .CV file. Internally, the Convert14 program is used.
`cartvte`	Export an ILWIS polygon or segment map to a Cart/o/graphix .VTE file. Internally, the Convert14 program is used.
`dbase`	Export an ILWIS table to a dBase III/IV file (.DBF).
`dbasesdf`	Export an ILWIS table to an ASCII dBase III/IV file (.SDF).
`delimited`	Export an ILWIS table to an ASCII comma delimited file (.TXT).
`dxf`	Export an ILWIS polygon, segment or point map to an AutoCad .DXF file.
`e00`	Export an ILWIS polygon, segment or point map to an Arc/Info .E00 file.
`erdasdig`	Export an ILWIS polygon or segment map to an Erdas .DIG file. Internally, the Convert14 program is used.
`erdasgis`	Export an ILWIS raster map to an Erdas .GIS file; in case you exported a domain Class map, also a trailer file will be created (.TRL). The ILWIS map should preferably be north-oriented.
`erdaslan`	Export an ILWIS map list, which contains multiple bands of a satellite image, or export a single ILWIS raster map to an Erdas .LAN file. The ILWIS map(s) should preferably be north-oriented.
`geosoft`	Export an ILWIS raster map to a Geosoft Grid file (.GRD). Internally, the Convert14 program is used.
`gif`	Export an ILWIS raster map to a GIF image (.GIF). Internally, the Convert14 program is used.
`gina`	Export an ILWIS segment map to a Gina file (.GIA). Internally, the Convert14 program is used.
`hpgl`	Export an ILWIS segment map to an HPGL file (.HPG). Internally, the Convert14 program is used.
`ida`	Export an ILWIS raster map to an IDA image (.IMG). Internally, the Convert14 program is used.

`idrisi`	Export an ILWIS raster map to an Idrisi map (.DOC, .IMG).
`ilwis14`	Export an ILWIS raster, polygon, segment, point map or table to an ILWIS 1.x raster, polygon, segment, point map or table (.MPD, .POL, SEG, PNT, TBL and others).
`infocam`	Export an ILWIS polygon, segment or point map to an InfoCam sequential file (.SEQ).
`lin`	Export an ILWIS segment map to an Arc/Info .LIN file (ASCII). From this file, lines can be created using the Arc/Info Generate command.
`pcx`	Export an ILWIS raster map to a PaintBrush image (.PCX). Internally, the Convert14 program is used.
`shapefile`	Export an ILWIS polygon, segment or point map to Arc/View Shape files (.SHP, .SHX, .DBF).
`sif`	Export an ILWIS segment map to an ASCII Intergraph file in Standard Interchange Format (.SIF). Internally, the Convert14 program is used.
`smt`	Export an ILWIS segment map to an ILWIS 1.x ASCII vector file (.SMT).
`themak2`	Export an ILWIS raster, polygon, or segment map to a Themak2 file (.THR, .THP, THS). Internally, the Convert14 program is used.
`tiff`	Export an ILWIS raster map to a Tiff image (.TIF).
`usemap`	Export an ILWIS polygon map to a UseMap file (.USE). Internally, the Convert14 program is used.

Examples:

```
export e00(water.mps, e00water)
```

Exports the ILWIS segment map WATER.MPS to the file E00WATER.E00 in the Arc/Info interchange format. This example also illustrates the need for the explicit mentioning of the ILWIS map extension: as the E00 export is defined for segments, polygons and points there is no other way than the extension to tell export which type should be exported.

```
export erdasgis(soil.mpr, erdsoil)
```

Exports the ILWIS raster map SOIL.MPR to the file ERDSOIL.GIS. If the ILWIS map has a class domain, ILWIS will also create a SOIL.TRL file.

For more information, see also the Export dialog box, or the `expras.def`, `expmpl.def`, `exppol.def`, `expseg.def`, `exppnt.def`, `exptbl.def` files in your ILWIS directory.

☞ Besides using the Export command in a script, you can also use it within ILWIS on the command line; then, the complete command must be followed by a semicolon.

Appendix F

Errata ILWIS 2.1 Reference Guide

p. 39	In the first tip ☞, the first line was left out. The complete tip is: You can increase or decrease the number of characters shown by Info in a map window and in the map editors by setting the 'width' in the properties of the domain. The default width for a class domain is 15; the default width for an identifier domain is 6.

p. 54 Replace: Also, when only the object definition file of a dependent object exists, … pixel info is able to …

With: **Special functionality**

When only the object definition file of a dependent map exists, i.e. when a dependent map has not been calculated yet and the data file of the dependent map does not yet exist, then pixel info is able to …

After paragraph "Working with dependent maps, … stored on disk."

Insert: Furthermore, when for value raster maps you marked the 'Interpolation' check box in the Properties dialog box of a raster map, the Pixel info window will show interpolated values on sub-pixel level. For more information, refer to the Raster Map Properties dialog box.

Finally, you can also add coordinate systems to the pixel information window. When a transformation is possible between the coordinate system of the current map and the added coordinate system, the pixel information window will show coordinates transformed to the new coordinate system, i.e. a transformation is performed on the fly. In this way you can already see the results of a transformation to another coordinate system before you actually perform this transformation, thus before using a Transform operation or the Resample operation, and you can compare different coordinate systems with each other.

For more information, see possible coordinate system transformations and an example of adding multiple coordinate systems in the pixel information window.

p. 131 Segment editor Menu commands:

In the File menu, insert above Polygonize: Remove Redundant Nodes

p. 163, p. 168, p. 209, App. 30

New arithmetic operator:

^	*a*^*b*	exponential operator; *a*^*b* = POW(*a*,*b*) = a^b

p. 164, p. 222, App. 34

Statistical functions on Value columns have been extended:

AVG(*col*)	average of column *col*
MIN(*col*)	minimum of column *col*
MAX(*col*)	maximum of column *col*
SUM(*col*)	sum of column *col*
STD(*col*)	standard deviation of column *col*
STDEV(*col*)	standard deviation of column *col*
STERR(*col*)	standard error of column *col*
VAR(*col*)	variance of column *col*
COV(*col1, col2*)	covariance of column *col1* and column *col2*
COVAR(*col1, col2*)	covariance of column *col1* and column *col2*
CORR(*col1, col2*)	correlation of column *col1* and column *col2*
TTEST(*TrueValue, col)*	
	Student's *t*-test on column *col* where *TrueValue* is the value to which the mean of your variable stochastically converges (m).
CHISQUARE(*col, colExpValue*)	
	C² -test on the observed values (f_i) in column *col* where *colExpValue* is the column with the expected values (F_i)

p. 164, p. 236, App. 35

New Conversion functions CODE(*s*) and NAME(*s*); changed description VALUE(*s*) function:

CODE(*s*)	returns the code of *s* as a string; *s* must have a Class or ID domain; when there is no code, an empty string is returned
NAME(*s*)	returns the name of *s* as a string; *s* must have a Class or ID domain
VALUE(*s*)	returns numbers in string *s* as a value

p. 183, p. 231, App. 31, App. 34

Calculating with undefineds:

ISUNDEF(*a*)	tests whether *a* is undefined.
IFUNDEF(*a,b*)	if condition *a* is undefined, then return the outcome of expression *b*, else return *a*.

IFUNDEF(*a,b,c*)	if condition *a* is undefined, then return the outcome of expression *b*, else return the outcome of expression *c*.
IFNOTUNDEF(*a,b*)	if condition *a* is not undefined, then return the outcome of expression *b*, else return undefined.
IFNOTUNDEF(*a,b,c*)	if condition *a* is not undefined, then return the outcome of expression *b*, else return the outcome of expression *c*.

p. 195 Reads: `IFF(Demnew > 3340, Start, NBMAX(Start1#))`
Should read: `IFF(Demnew > 3340, Start, NBMAX(Start#))`

p. 227

INMASK(*s*, "*mask*")	tests whether *s* conforms to one of the strings specified in the "*mask*". Argument *s* can be a column name using domain type class, ID, group or string, or the outcome of an expression using domain type class, ID, or group or a string. In the mask, you can specify multiple search strings, each separated by a comma, and you can use wildcards ? and *. The total mask has to be enclosed by double quotes.

p. 228 Reads: `Result3=IFF( …,(”not suitable”)`
Should read: `Result3=IFF( …,”not suitable”)`

p. 243 Reads: `? MAPVALUE(tmb1, coord(800000,8080000),cochabam)`
Should read: `? MAPVALUE(tmb1, coord(800000,8080000,cochabam))`

p. 248 Reads: `Altitude = MapValue(DEM.mpr, PNTCRD(COORD(X,Y)))`
Should read: `Altitude = MapValue(DEM, COORD(X,Y))`

p. 270: Pictures of input and output map should be switched.

p. 278, 280, 281
By specifying a Row Offset and/or a Column Offset, the operation does not *start* from the specified row or column number but the specified number of rows and columns are *skipped*.

p. 282-289
In all topics on Distance calculation, insert:
In the Distance calculation dialog box, a source map can be any raster map with a class or ID domain. On the command line, you can also use raster maps with a value domain.

p. 337 Reads: `Hue = 255/PI * …`
Should read: `Hue = 255/2PI * …`

p. 465 Under Dialog box options, after ‘Input point map’, insert:

Attribute:	In case the input point map is linked to an attribute table, select this check box and select an attribute column from the attribute table linked to obtain a raster map from the attribute values of the points.

p. 466 At the top of this page, after ‘Georeference’, insert:

Value range:	In case the input and output map use a value domain, accept the default value range, or specify your own range of possible values in the output map.
Precision:	In case the input and output map use a value domain, accept the default precision of output values, or specify your own precision.

p. 516 Replace the explanation under point 5 with:
To calculate a slope map in percentages from these maps DX and DY, type on the command line of the Main window:

```
SLOPEPCT = 100 * HYP(DX,DY)/ PIXSIZE(DEM)
```

HYP is an internal Mapcalc/Tabcalc function.
Function PIXSIZE returns the pixel size of a raster map; for DEM, fill out the name of your DEM created in step 2.
SLOPEPCT is the output map name of the slope map in percentages.

p. 563 How to edit maps

Replace:	In a polygon map, new polygons can be inserted and existing ones can be deleted, codes of polygons can be edited, the shape of a polygon boundary can be changed and unnecessary polygon boundaries can be deleted;
With:	In a polygon map, the class names, IDs, or values of selected polygons can be edited, labels can be created from polygons, labels can be applied to polygons and segment can be extracted from polygons;

p. 577

First formula:	`Landuse.hsp.Area`
Should read:	`Landuse.hsa.Area`

App. 34, 35
In section For maps and columns of domain Class, ID, or Group, insert:
INMASK(*s*, "*mask*")
For an explanation of this function, see above.

App. 34

Reads:	IN *s2* IN *s1*;	tests whether *s2* is part of *s1*
Should read:	IN(*s1,s2*)	tests whether *s2* is part of *s1*

Appendix G

Software License Agreement

The International Institute for Aerospace Survey and Earth Sciences (ITC), P.O. Box 6, 7500AA, Enschede, The Netherlands (hereinafter referred to as ITC) herewith states that the use of the software program on the enclosed CD is subject to the terms of this ILWIS License Agreement. You should not open this packet until you have read the terms written below. By opening the packet, you signify that you have read this agreement and accept its terms.

1. DEFINITIONS
'Licensed Software' shall be the product containing the software - in whole or in part - as described on the Order Form and any updates furnished by ITC in connection with the licensed Software, including basic and related materials pertinent to the said software. 'Derivative Software' shall be the 'Licensed Software' which has been merged into other software materials to form a 'Derivative Work'.

2. LICENSE
The license granted under this Agreement authorizes the Licensee on a non-exclusive and non-transferable basis to use the Licensed Software and any Derivative Work only within and on behalf of the Organisation as described above.

3. RESTRICTION
Licensee agrees not to provide or otherwise make available the Licensed Software and any Derivative Work to any person other than Licensee's employees. A separate license - to be obtained through ITC - is required for use of the Licensed- and any Derivative Software by other Organisations.
Licensee agrees to take appropriate action with its employees to satisfy Licensee's obligations under this Agreement with respect to copying, protection and security of the Licensed Software.

4. TITLE
Title to and ownership of the Licensed- and Derivative Software and copies thereof shall at all times remain with ITC.

5. OBLIGATIONS
Licensee agrees that with the delivery of the Licensed Software ITC enters no obligations whatsoever.

6. LIABILITY
Licensee agrees that ITC shall not be liable for any loss, claim or damage arising out of Licensee's possession or use of the Licensed- and Derivative Software.

7. TERM
This Agreement is effective as per breaking the seal and shall remain in force for an indefinite period of time.

8. TERMINATION
Licensee agrees that with the termination of this Agreement the Licensed- any Derivative Software and all copies thereof shall be destroyed..

9. GOVERNING LAW
This Agreement is governed by Dutch law.

Appendix H

Customers Services Report Form

Please send a copy of the form to:

ITC
ILWIS Help desk
P.O. Box 6
7500 AA Enschede
The Netherlands
Fax: +31 53 4874484
E-mail: support.ilwis@itc.nl

Key:..
License:...
Version:...

Name..
Organization:...
Address:...
Country:...
Telephone:..
Fax:...
E-mail:...

Define your problems :
Problem type:

- ☐ Computer reboots
- ☐ Computer crashes
- ☐ Program quits at unexpected moment
- ☐ Error message, namely :..................................
- ☐ Incorrect results

Problem frequency:

- ☐ Regularly
- ☐ Irregularly
- ☐ Data related
- ☐ In connection with:...

Hardware used and platform type:...
....
....
....
....
....
....

Operations in use:...
....
....
....
....
....
....
....
....

Specify procedure in detail:...
....
....
....
....
....
....
....
....

Expected results:...
....
....
....
....
....
....
....

What happened, please specify in detail:...
....
....
....
....
....
....
....
....