CHAPTER 12

Advanced spatial data analysis: command line, functions and scripts

In the previous chapters you have seen a number of examples of data analysis, which were mainly performed by using dialog boxes of the various ILWIS operations.For more advanced users, spatial data analysis in ILWIS can also be done in a more rapid way, by typing a *command* or an *expression* on the command line of the Main window. By typing a command of a certain operation, the dialog box of that operation is opened, so you can work faster then by selecting menu items. *Commands* are also used for all kinds of data management operations, like copying, deleting, changing directories, and dealing with dependency links. In an *expression* all the parameters which you would normally enter in the dialog box have to be specified. In the first part of this chapter you will learn how to use the command line.

In chapter 5 (attribute data handling) and chapters 7 to 9, you have been working with calculation formulas that either work with tables, or with maps. Instead of typing these formulas each time once again, you can also store frequently used formulas as *functions*, which can then be used for different maps by including parameters in the function. The use of functions will be treated in the second part of the chapter.

The final part of the chapter deals with the use of *scripts*. A script contains a list of commands and expressions. You can use scripts to automate your analysis. Sequences of operations can be executed automatically with a script. A script is comparable with the use of batch files in previous ILWIS DOS versions.

Before you can start with the exercises, you should start up ILWIS and change to the subdirectory c:\ilwis21\data\usrguide\chap12, where the data files for this chapter are stored.

☞

- Double-click the ILWIS program icon in the ILWIS program group.
- Change the working drive and the working directory until you are in the directory c:\ilwis21\data\usrguide\chap12.

12.1 Working from the command line

Most of the activities that are done via menus can also be done via the command line of the Main window, located at the top of the Main window, just below the menu bar. A picture of the Main window is presented in Help topic Main window : Introduction.

The command line is used for the following activities:

– To perform *calculations* with maps and attribute tables. The calculations with tables were treated in chapter 5, and the Map Calculation statements were discussed in chapter 8;.

– To perform *ILWIS commands*: To display, edit, create or view the properties of ILWIS objects, and to obtain dialog boxes to start an ILWIS operation. This functionality is equivalent to typing an ILWIS 1.4 executable name on the DOS command line in ILWIS version 1.4. You can also use ILWIS commands for *data management*: copying and deleting objects, and breaking dependency links of objects;

– To perform *ILWIS expressions* allowing you to perform complete ILWIS operations directly from the command line. This functionality is equivalent to typing the ILWIS 1.4 executable name and all parameters required by this executable, on the DOS command line in ILWIS version 1.4;

– To run *scripts*, which usually contain a sequence of ILWIS expressions. With a script, you can build a complete GIS and Remote Sensing analysis/application for your own research discipline. Scripts are more or less equivalent to batch files in ILWIS version 1.4;

– To use it as a *pocket calculator*. If you put a question mark before an expression, the result of that expression is shown on the command line. For example: When you type ?100/5 ↵ , a message box will show the value 20;

– To *start any Windows application program*, batch file, or DOS application (with a .PIF file available), by writing an exclamation mark in front of the application name. Applications that can be started from the command line may have the following extensions: .EXE, .COM, .BAT, .PIF. Type the application name directly after the exclamation mark (no spaces allowed). For example, to start Word, type: !Winword.

! The command line has a history: Use the ARROW UP key to retrieve previously used expressions and commands.
You can also copy and paste text back and forth from the command line to the clipboard with the following keystrokes:

Ctrl+C	Copy the selected part to the clipboard.
Ctrl+V	Paste the contents from the clipboard.

12.1.1 Commands for opening or editing an ILWIS object

To open an ILWIS object you can type one of the commands listed in table 12.1. Since the command line is case insensitive, it doesn't matter if you type them in upper or lowercase.

Table 12.1: The commands that can be used on the command line for opening objects

Command	Example	Description
Open	Open	Displays the Show Map dialog box from which you can select the object to be opened.
Open `object.ext`	Open `Geom.mpr`	Opens the Display Options dialog box for the object or shows the object immediately (in the case of tables for example).
Show	Show	Same as Open : displays the Show Map dialog box from which you can select the object to be opened.
Show `object.ext`	Show `Landuse.tbt`	Opens the Display Options dialog box for the object or shows the object immediately (in the case of tables for example).
Map `rastermapname`	Map `Landuse`	Gives the Display Options dialog box for the raster map.
Pol `polygonmapname`	Pol `Citybl`	Gives the Display Options dialog box for the polygon map.
Seg `segment map name`	Seg `Contour`	Gives the Display Options dialog box for the segment map name.
Pnt `pointmapname`	Pnt `Rainfall`	Gives the Display Options dialog box for the point map name.
View `viewname`	View `Cocha`	Displays the view in a map window.
Mpl `maplistname`	Mpl Tms	Displays a maplist as a slide show.
Tbl `tablename`	Tbl `Citybl`	Displays the table in a table window.

! Users that are familiar with the use of the command line in earlier DOS versions of ILWIS, can also use the names used in ILWIS 1.41 for opening raster maps and tables. These commands are called *Aliases*:

CopyMap Alias for the command Map.
TabCalc Alias for the command Tbl.

☞

- Type the following command on the command line:
  `open ↵`
- The Show map dialog box appears from which you can select any object.
- Select the raster map `Geom` and click OK. The Display Options dialog box is opened. Click OK. The raster map `Geom` is opened.
- Close the map window.
- Type the following command on the command line:
  `open Geom ↵`

You will notice that nothing happens when you type: `open Geom ↵`

This is due to the fact that ILWIS doesn't know which object with the name `Geom` should be displayed: A raster map, polygon map or table. So with the open command you have to use the name of the object and the extension.

☞

- Type the following command on the command line:
  `open Geom.mpr↵`

The extension `.mpr` is the extension of raster maps. In table 12.2 the file extensions of the different map types and tables are listed.

Table 12.2: File extensions of different map types and of tables

Extension	**Example**	**Description**
.mpr	Geom.mpr	File extension of raster maps.
.mpa	Citybl.mpa	File extension of polygon maps.
.mps	Contour.mps	File extension of segment maps.
.mpp	Rainfall.mpp	File extension of point maps.
.mpv	Cocha.mpv	File extension of map view.
.mpl	Tms.mpl	File extension of a map list.
.tbt	Citybl.tbt	File extension of tables.

☞

- Click the Cancel button in the Display Options dialog box.
- Type the following command on the command line:
  `show Geom.mpr↵`
- The command Show has exactly the same function as the command Open. Click the Cancel button in the Display Options dialog box.
- Type the following command on the command line:
  `map Geom ↵`
- The command map is used for opening raster maps, so you do not need to put the extension `.mpr`. Click the Cancel button in the Display Options dialog box.
- Type the following command on the command line:
  `Copymap Geom.mpr↵`

☞

- The command Copymap is the old ILWIS 1.41 command that was used to display raster maps. You can still use it in the new version, as a so-called *alias*. The command Copymap has the same effect as Open or Show. Click the Cancel button in the Display Options dialog box.
- Practice some more with the other commands used for displaying maps and tables (shown in table 12.1), such as:

  `show Landuse.tbt ↵`

  `map Landuse ↵`

  `pol Citybl ↵`

  `seg Contour ↵`

  `pnt Rainfall ↵`

  `view Cocha ↵`

  `tbl Citybl↵`

  `mpl Tms ↵`
- Close all map and table windows when finished.

It is also possible to open the pixel information window from the command line by typing : `Pixelinfo ↵`

If you want to edit an ILWIS object : Type `Edit` and an Edit Object dialog box appears, in which you can select an object to be edited.

☞

- Type the following command on the command line:

  `edit ↵`
- The Edit object dialog box is opened. From this dialog box you can select the object to be edited. From the drop-down list box in the lower part of the dialog box you can select which type of objects you want to list in the list box. On the right hand side you can select the directory and the drive. Click the Cancel button.
- Type the following command on the command line:

  `Edit Geom.mpa↵`
- The Display Options dialog box for the polygon map `Geom` is opened. Click OK. The Polygon editor is opened.
- Close the map window.

You can also use the command line to see and change the properties of objects. The properties of maps and tables were described in chapter 2. If you want to see or change the properties, type: `Prop` and the Edit Properties dialog box appears, in which you can select an object of which you want to see the properties.

☞

- Type the following command on the command line:
  `prop↵`
- The View properties of... dialog box is opened. From this dialog box you can select an object of which you want to see the properties. Click the Cancel button.
- Type the following command on the command line:
  `prop Geom.mpa↵`
- The Polygon map properties dialog box is opened. From this dialog box you can see the properties of the polygon map `Geom`, and you can change the properties, as described in chapter 4. Click the Cancel button.

12.1.2 Commands for creating ILWIS objects

Apart from opening and editing objects, we can also use commands on the command line to open the dialog boxes dealing with the creation of objects. A number of commands for creating objects is shown in table 12.3.

Table 12.3: Some commands that can be used on the command line for creating objects

Command	Description
Create map	Gives the Create Raster Map dialog box.
Create seg	Gives the Create Segment Map dialog box.
Create pnt	Gives the Create Point Map dialog box.
Create tbl	Gives the Create Table dialog box.
Create dom	Gives the Create Domain dialog box.
Create rpr	Gives the Create Representation dialog box.
Create grf	Gives the Create Georeference dialog box.
Create csy	Gives the Create Coordinate system dialog box.
Create mpl	Gives the Create Maplist dialog box.
Create fil	Gives the Create Filter dialog box.
Create fun	Gives the Create Function dialog box.
Create isl	Gives the Create Script dialog box.

☞

- Type the following command on the command line:
  `Create dom ↵`
- The Create domain dialog box is opened. In this dialog box you have to enter the name of the domain and the domain type. Since this was just to show you how to obtain the dialog box from the command line click the Cancel button.

The commands given above only open a certain dialog box. They do not create a certain object by themselves. It is also possible to do that directly from the command line. A number of examples are shown in table 12.4. Whereas the examples shown in the previous tables are *commands*, the examples in table 12.4 are *expressions*.

By typing an ILWIS *command* on the command line of the Main window, a dialog box is usually opened. By typing an ILWIS *expression* on the command line, the operation's dialog box is skipped. All parameters required by an operation have to be entered.

Table 12.4: Some examples of expressions that can be used on the command line for creating objects

Command	Example	Description
crdom	crdom xnewdom1 -type=class -items=10 -prefix=cl ;	Creates a class domain with name “xnewdom1” and ten items that have class names “cl1”, “cl2”,...”cl10” .
crdom	crdom xnewdom2 -type=class ;	Creates a class domain with name “xnewdom2” without any items.
crdom	crdom xnewdom3 -type=id -items=100 -prefix=prov ;	Creates an identifier domain with name “xnewdom3” and 100 items that have class names “prov1”, “prov2”, “prov3”,...”prov100”.
crdom	crdom xnewdom4 -type=value -min=100 -max=200 ;	Creates a value domain with name “xnewdom4”, value range 100-200 (precision is 1).
crdom	crdom xnewdom5 -type=value -min=-10 -max=200 -prec=0.01 ;	Creates a value domain with name “xnewdom5”, value range -10 to 200, and precision 0.01.
additemtodomain	additemtodomain xnewdom2 “forest” ;	Adds the item “forest” to the class domain “xnewdom2”.
additemtodomain	additemtodomain xnewdom3 “prov101” ;	Adds the item “prov101” to the identifier domain “xnewdom3”.
crrpr	crrpr xnewcol xnewdom1 ;	Creates the representation “xnewcol” for the domain xnewdom1.
crgrf	crgrf xcoch2 500 1000 -crdsys=cochabam -lowleft=(796000,8072000), -pixsize=10;	Creates a georeference with the name “xcoch2”, with 500 rows and 1000 columns and coordinate system Cochabam. It has as lower left coordinate (796000,8072000) and as pixel size 10 m.

! When you type an ILWIS expression on the command line, the expression should end with a semicolon (;). This makes sure that the expression is executed without further involvement of the user interface (so without any dialog boxes shown).

To practice with some of these expressions, you will now create a domain and add items to it.

☞

- Type the following command on the command line:
  `crdom Xnewdom2 -type=class ; ↵`
- The domain `Xnewdom2` is created, but it still doesn't contain any items.
- Type the following command on the command line:
  `additemtodomain Xnewdom2 “forest”; ↵`
  `additemtodomain Xnewdom2 “no forest”; ↵`
- The domain `Xnewdom2` now has two items. Check this by opening the domain `Xnewdom2`.
- Practice also with the other examples shown in table 12.4. Before you finish, make sure to close all open windows and dialog boxes, except for the Main window.

The expression as shown in table 12.4 can be used on the command line, as well as in scripts. This will be further explained in exercise 12.4.

12.1.3 Commands used for Data Management

There are also a series of commands that you can use for data management. They are listed in table 12.5.

Table 12.5: Some examples of commands and expressions that can be used on the command line for data management

Command	Example	Description
copy	copy	Opens Copy object dialog box.
copy object.ext	copy Geom.dom	Opens Copy object dialog box with the name of the object already listed.
copy object.ext newobject.ext;	copy Geom.dom Xgeom.dom;	Copies the specified object to the new object.
copy object.ext directory ;	copy Geom.dom c:\ilwis21\data	Copies the specified object to the specified directory.
copyfile object.ext directory;	copyfile Geom.dom c:\ilwis21\data	Copies the specified object to the specified directory.
copy object.ext directory\newobject.ext;	copy Geom.dom c:\ilwis21\data\usrguide\Xgeom.dom	Copies the specified object to the new directory with the given name.
del	del	Opens the Delete object dialog box.
del object.ext;	del Xtra.dom;	Deletes the specified object.
cd\	cd\	Goes to the root of the current drive.
cd\dir	cd\ilwis21\chap111	Goes to the specified (sub)directory.
cd..	cd..	Goes one directory up.
cd subdir	cd test	goes to the subdirectory.
md	md	Opens the dialog box to create a new subdirectory.
md newdir	md test	Creates a new subdirectory.
mkdir newdir	mkdir test1	Creates a new subdirectory.
rd	rd	Opens the dialog box to remove a subdirectory.
rd subdir	rd Test1	Removes the subdirectory.
rmdir subdir	rmdir Test1	Removes the subdirectory.
a:	a:	Goes to the specified drive.
exit	exit	Leaves ILWIS.

With the group of commands and expressions shown in table 12.5 you can copy and delete files, and create, change or remove directories.

☞

- Type the following command on the command line:

  `md Test ↵`

  `cd Test ↵`

  Now you are in the directory c:\ilwis21\data\usrguide\chap12\test.

  `cd.. ↵`

- Now you are in the directory c:\ilwis21\data\usrguide\chap12.

Since the catalog of the Main window displays only the files from one (sub)directory at a time, the commands for changing directories are quite important.

It is advised to keep all files that are linked within the same (sub)directory. So the coordinate system, the georeference, the domain and the representation, should be stored in the same subdirectory as a raster map, which uses these objects. Also, for dependent data objects it is recommended to keep them in the same (sub)directory. This is important, since the directory of the service object files is stored in the properties of a data object. If service objects are stored in different (sub)directories, this may cause problems when copying files.

Copying of files can be done via the Copy Object dialog box, or directly via the command line. When you use the Copy Object dialog box, you can only copy one object at a time. All the service objects needed for the data object will also be copied. If the object is dependent, the source objects will also be copied. To prevent the copying of the source objects also, you can first break the dependency link of the dependent data object.

Copying of ILWIS files with the Windows File Manager or any other program is not advised, unless you know which service and other data objects are linked to the file you want to copy. It is easy to overlook a georeference, which is used by the raster map that you want to copy, but may have another name. If you use ILWIS itself for copying files, the program will also copy all service objects that are needed.

When you copy files via an expression on the command line, you can copy more objects at a same time, using *wildcards (* and ?)*. Some examples are given below.

☞

- Type the following command on the command line:
  `copy Geom.dom Xgeom.dom;` ↵
- Check in the data catalog whether there is a new domain file Xgeom.
- Type the following command on the command line:
  `copy Geom.* c:\ilwis21\data\usrguide\chap12\test;` ↵
  `cd test` ↵
- Now all the files with the name Geom and any extension are copied to the subdirectory c:\ilwis21\data\usrguide\chap12\test.

If you want to copy files from the command line, without taking into account its dependency links, you can use the Copyfile command on the command line.

Deleting of files can be done via the Delete object dialog box, or directly via the command line. When you use the command line, you can either type Del or Delfile. Del is an internal ILWIS command, and the program will check the dependency links of the object that you want to delete. When you want to delete files, using wildcards, without taking into account the dependency links, you can use the command Delfile.

! **Warning:**
You should be careful with the use of the delfile command from the command line, since it can be used to delete objects which are essential for others. For example, if you delete a domain, you cannot open the object (maps or tables) that use this domain. Therefore, you should first examine if the object(s) that you want to delete is/are not used by other objects. This can be done via the Used by button in the Properties dialog box of an object.

☞

- Make sure that you are in the subdirectory c:\ilwis21\data\usrguide\chap12\test
- Type the following command on the command line:
  `Del Cochabam.grf ↵`
- Now a warning appears that the georeference Cochabam is used by other objects. `Delete anyway?` Click No.
- Type the following command on the command line:
  `delfile Cochabam.grf ↵`
- The georeference `Cochabam` is now deleted.
- Open the raster map `Geom`.
- An ILWIS error message will appear indicating:
  `c:\ilwis21\data\usrguide\chap12\test\Cochabam.grf Find error`.
- Click OK. The Display Options dialog box appears. Click OK. The raster map is now displayed.
  Note that the map no longer has coordinates, since these were coming from the deleted georeference file `Cochabam`. So be sure of what you delete.
- Type the following command on the command line:
  `delfile Geom.* ↵`

☞

- All the files names Geom with different extensions are now deleted.
- Type: `cd ..` ↵ You are back again in the directory c:\ilwis20\data\usrguide\chap12
- Type the following command on the command line:
  `Delfile x*.*;` ↵
- All the files that you created in the earlier part of the exercise, all of which start with an `x`, are now deleted.

12.1.4 Commands used to get Help

You can open the ILWIS Help from the command line by typing: `Help` ↵

You can also open the Search dialog box by typing Help with a search string.

☞

- Type the following command on the command line:
  `Help` ↵
- The Contents page of the ILWIS Help is opened .
- Click the Cancel button.
- Type the following command on the command line:
  `Help operations`↵
- The Topics Found dialog box of the ILWIS Help is opened. Select ILWIS Operations. Click the Display button.
- Click the topic ILWIS operations quick reference. You are now in the part of Help that gives an overview of all ILWIS operations, ordered in the same way as in the menu of the Main window.
- Click any of the operation names to see a short description of the operation.
- Close the Help.

12.1.5 Commands and expressions for ILWIS operations

Up to now we have seen only commands and expressions that are useful for opening, editing, copying and deleting objects. The command line can also be used to open a dialog box of an ILWIS operation. The list of commands for the various operations is shown in table 12.6. The table shows the *commands* and the *aliases*. Aliases are the ILWIS 1.4 executable names. An alias is listed when the command is not the same as the ILWIS 1.4 executable name.

The last column of table 12.6 shows examples of how you can perform the operation from the command line with the help of an *expression*, in which all the parameters needed for the operation are specified. By typing an ILWIS expression on the command line, the operation's dialog box is skipped. You can for instance create a color composite, filter a map or perform a cross.

The general syntax for expressions is:

```
OUTMAP = expression
```

```
OUTMAP := expression
```

– The name of the output object (e.g. OUTMAP).

– A definition symbol (=) or an assignment symbol (:=) to indicate whether to create a dependent output object, or a source object.

– An expression consisting of an operation name followed by parameters (between brackets, and separated by commas), or a MapCalc expression.

ILWIS expressions are not case-sensitive. Spaces are allowed after output object names, after operation names, and before and behind brackets, commas, or semi-colons. Spaces are not allowed before or behind colons.

Any operation name in the table 12.6 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.

☞

- Open the ILWIS Help.
- Click the Search button in the Help window
- The Search dialog box of the ILWIS Help is opened.

- Type the search item: `expressions`.
- Click the Display button.
- Select the topic Appendices: ILWIS expressions and click Display. You are now in the part of Help that gives an overview of all ILWIS expressions.
- Scroll through the list until you come to the topic Attribute in the Raster Operations.
- Click the hypertext link MapAttribute.

  The Help now moves to the topic Attribute map of Raster Map.
- Read the Help on this topic and close the Help afterwards.

You will work with an example dealing with the reclassification of a raster map (`Landuse`) using the data from a column of its attribute table. The average land prices per hectare are given in the column `Landvalue`.

☞

- Open the raster map `Landuse` and the table `Landuse`, and check their contents.
- Type the following command on the command line:

  `Attribras ↵`
- The Attribute Map dialog box is opened.

In this dialog box you will have to specify the parameters for the operation. This is an example of using *commands* to open the dialog boxes of operations from the command line. If you want to create the result map in one statement, you will have to use an *expression*, in which the parameters are already given.

☞

- Close the dialog box Attribute Map by clicking the Cancel button.
- Close the map `Landuse` and the table `Landuse.`
- Type the following expression on the command line:

  `Landval=MapAttribute(Landuse,Landvalue); ↵`

Table 12.6: Overview of commands and expressions that can be used on the command line for ILWIS operations

Operation	Command	Example of expression
Visualization		
Show Map	Show	**show** object.ext
Show Table	Tbl	**tbl** table.tbt
Color Composite	ColorComp	OutMap = **MapHeckbert**(map list, nr of colors)*
Display 3D	Display3d	no expression available: interactive process
Apply 3D	Apply3d	OutMap = **MapApply3d**(rasmap, georeference3D)
Slideshow	Mpl	no expression available
Raster Operations		
Map Calculation	MapCalc	OutMap = map calculation expression (see chapter 7 to 9)
Attribute Map	Attribras	OutMap = **MapAttribute**(rasmap,*tbl*.column)
Cross	Cross	OutMap = **MapCross**(rasmap1, rasmap2, *output cross table*)
Distance calculation	Distance	OutMap = **MapDistance**(source rasmap,*weight map*)
Area numbering	AreaNumb	OutMap = **MapAreaNumbering**(rasmap,*8/4*)
SubMap	SubMap	OutMap = **MapSubMap**(rasmap,first row,first col,nr rows,nr columns)
Glue Maps	Glueras	OutMap = **MapGlue**(rasmap1, rasmap2,*replace*)
MirrorRotate	Mirror	OutMap = **MapMirrorRotate**(rasmap, *rotate type*)
Image processing		
Filter	Filter	OutMap = **MapFilter**(rasmap, *filter*)
Stretch	Stretch	OutMap = **MapStretchLinear**(rasmap, range from, domain)*
Slicing	Slicing	OutMap = **MapSlicing**(rasmap, domain group)
Color Separation	Colorsep	OutMap = **MapColorSep**(rasmap, color)
Cluster	Cluster	OutTable = **MapCluster**(maplist,nr of colors)
Sample	Sample	no expression available: interactive process
Classify	Classify	OutMap = **MapClassify**(sample set, *classifier*)
Resample	Resample	OutMap = **MapResample**(rasmap, georef, *resampling method*)
Statistics		
Histogram	Histogram	OutTable = **TableHistogram**(rasmap) *
Statistics raster		
Autocorrelation	Autocorr	OutTable = **TableAutoCorrSemiVar**(pntmap, *max shift*)
Statistics Maplist		
Principal Components	PrincCmp	OutMap = **MatrixPricComp**(map list)
Factor Analysis	FactAnal	OutMap = **MatrixFactorAnal**(map list)
Variance Covariance	MatVarCov	no expression available
Correlation Matrix	MatCorr	no expression available
Statistics Polygons		
Neighbour Polygons	HistNbPol	OutTable = **TableNeighbourPol**(polmap)
Statistics Segments		
Directional Histogram	HistSegDir	OutTable = **TableSegDir**(segmap)
Statistics Points		
Spatial Correlation	SpatCorr	OutTable = **TableSpatCorr**(pntmap,*column*)
Pattern Analysis	PattAnal	Outtable = **TablePattAnal**(pntmap)
Interpolation		
Densify Map	DensRas	OutMap = **MapDensify**(rasmap, factor,interpolation method)
Contour interpolation	Interpolseg	OutMap = **MapInterpolContour**(segmap, georef)

Table 12.6 (continued): Overview of commands and expressions that can be used on the command line for ILWIS operations

Operation	Command	Example of expression
Point interpolation		
Nearest point	NearestPnt	OutMap = **MapNearestPoint**(pntmap,georef)
Trend surface	TrendSurface	OutMap = **MapTrendSurface**(pntmap,georef,*surface type*)
Moving average	MovAverage	OutMap = **MapMovingAverage**(pntmap,georef,*weight funct*)
Moving surface	MovSurface	OutMap = **MapMovingSurface**(pntmap,georef,*surface*,*weight fun*)
Vector operations		
UniqueID	UniqueID	OutMap = **PolygonMapNumbering**(polmap) *
Polygons:Attribute	AttribPol	OutMap = **PolygonMapAttribute**(polmap,*tbl*.column)
Polygons: Mask	MaskPol	OutMap = **PolygonMapMask**(polmap,”mask”)
Polygons: Label	LabelPol	OutMap = **PolygonMapLabels**(polmap,pntmap)
Polygons: Transform	TransfPol	OutMap = **PolygonMapTransform**(polmap,coord sys)
Segments: Attribute	AttribSeg	OutMap = **SegmentMapAttribute**(segmap,*tbl*.column)
Segments: Mask	MaskSeg	OutMap = **SegmentMapMask**(segmap,maks)
Segments: Label	LabelSeg	OutMap = **SegmentMapLabels**(segmap,pntmap,*set domain*)
Segments: SubMap	SubSeg	OutMap = **SegmentMapSubMap**(segmap,minX,minY, maxX, maxY)
Segments: Glue maps	GlueSeg	OutMap = **SegmentMapGlue**(*minX, minY, maxX, maxY*, segmap1, mask1, segmap2, mask2, ...)
Segments: Densify	DensSeg	OutMap = **SegmentMapDensifyCoords**(segmap, distance)
Segments: Tunelling	TunnelSeg	OutMap = **SegmentMapTunelling**(segmap, tunnel width, *remove*)
Segments: Clean	CleanSeg	OutMap = **SegmentMapClean**(segmap)
Segments: Transform	TransfSeg	OutMap = **SegmentMapTransform**(segmap,coord sys)
Points: Attribute	AttribPnt	OutMap = **PointMapAttribute**(pntmap, *tbl*.column)
Points: Mask	MaskPnt	OutMap = **PointMapMask**(pntmap, “mask”)
Points: SubMap	SubPnt	OutMap = **PointMapSubMap**(pntmap, minX, minY, maxX, maxY)
Points: Glue maps	GluePnt	OutMap = **PointMapGlue**(*minX, minY,maxX, maxY*, pntmap1, mask1, pntmap2, maks2,...)
Points: Transform points	TransfPnt	OutMap = **PointMapTransform**(pntmap, coord sys)
Transform coordinates	Transform	No expression available: interactive process
Rasterize		
Polygon to Raster	PolRas	OutMap = **MapRasterizePolygon**(polmap,georef)
Segment to Raster	SegRas	OutMap = **MapRasterizeSegment**(segmap,georef)
Segment Density	SegDens	No expression available
Point to Raster	PntRas	OutMap = **MapRasterizePoint**(pntmap,georef, *point size*)
Point Density	PntDensity	OutMap = **MapRasterizePointCount**(pntmap, georef, *point size*)
Vectorize		
Raster to Polygon	RasPol	OutMap = **PolygonMapFromRas**(rasmap, 8/4, smooth/nosmooth)
Raster to Segment	RasSeg	OutMap = **SegmentMapFromRasAreaBnd**(rasmap, 8/4, smooth/nosmooth)
Raster to Point	RasPnt	OutMap = **PointMapFromRas**(rasmap)
Polygon to Segment	PolSeg	OutMap = **SegmentMapPolBoundaries**(polmap)
Polygon to Point	PolPnt	OutMap = **PointMapPolLabels**(polmap)
Segment to Polygon	SegPol	OutMap = **PolygonMapFromSegment**(segmap, *class/id*)
Segment to Point	SegPnt	OutMap = **PointMapSegCoords**(segmap) *
Transpose table	transpose	OutTable = **TableTranspose**(table, column dom, *column, min:max*)
Import/export		
Import from 1.4	Import14	Import14 file14.ext *outputdir*
Import	Import	No expression available
Export to 1.4	Export14	Export14 object20.ext name14
Export	Export	No expression available

No dialog box appears now, but the map definition of the map `Landval` is stored in the Catalog. Check it. Note that you have used the *definition symbol* (=) and not the *assignment symbol* (:=). The result of this is that the map `Landval` is a dependent map. The next thing you would like to know, for example, is the number of pixels in the map with different land values. For that you need to calculate the histogram of the map `Landval`.

☞

- Type the following expression on the command line:
  `Landval=TableHistogram(Landval) ↵`

Note that the histogram definition is stored but not yet calculated, since you used the definition symbol.

☞

- Double-click the histogram `Landval`. Note that first the map `Landval` is calculated, before the histogram is calculated and displayed. Note the range of values.

Now it would be nice to classify the map `Landval` into three classes (high, medium and low) using the Slicing operation. For that we need to have a group domain. This could be created with the expression (`crdom`) explained earlier on, but we have already provided you with a group domain (`Landvcl`). This domain contains three classes: low (value below 100), moderate (100-500) and high (500-1000).

☞

- Type the following expression on the command line:
  `Landvcl=MapSlicing(Landval,Landvcl); ↵`

The map `Landvcl` is a dependent class map, and its definition is now stored in the catalog.

☞

- Double-click the map `Landvcl`. The map is calculated first. The Raster Map Display Options dialog box is opened.
- Click OK. Close the map window

Finally, suppose we want to cross the map `Landvcl` with the map `Geology`, in order to find out the landvalues for different geological units. We only want a cross table, not a resulting cross map.

☞

- Type the following expression on the command line:
  `Geolvcl=TableCross(Geology,Landvcl);` ↵
- Open the table `Geolvcl` and look at the result. Close it after that.

As you can see, the use of expressions for executing operations from the command line is quite convenient. In practice you will need to be quite familiar with the syntax of the various expressions, in order to benefit from it. The expressions are especially intended for the use in scripts, which are objects containing a series of expressions, that can be used to automate an analysis. Scripts will be treated later in this chapter.

12.1.6 Using Map and Table Calculation expressions

In chapter 8 you have seen the use of the command line for entering Map Calculation formulas. Table calculation formulas were treated in chapter 5. These have to be entered in the command line of the table window. However, it is also possible to enter them in the command line of the Main window.

When you normally enter a Map Calculation formula, such as:

`Map3 = Map1 + Map2,`

or a Table calculation formula, such as:

`Column3 = Column1 + Column2,`

you will see the Raster Map Definition dialog box (for Map Calculation) or a Column Properties dialog box (for Table calculation). In these dialog boxes you have to specify the domain of the output map or column.

It is also possible to skip these dialog boxes, by typing the definition of the output map or column directly in the expression. Some examples are given below for value domains:

```
Map3{dom=valuedomainname;vr=min:max:precision} = Map1+ Map2
Col3{dom=valuedomainname;vr=min:max:precision} = Col1+ Col2
Map3{dom=valuedomainname;vr=min:max} = Map1+ Map2
Col3{dom=valuedomainname;vr=min:max} = Col1+ Col2
Map3{vr=min:max;dom=valuedomainname} = Map1+ Map2
Col3{vr=min:max;dom=valuedomainname} = Col1+ Col2
Map3{dom=valuedomain;vr=::precision} = Map1 + Map2
Map3{dom=valuedomain;vr=::precision} = Map1 + Map2
```

If the result of the expression is a class or ID domain you can specify this also, for example:

```
Map3 {dom=classdomainname} = iff(Map1="Forest",Map1,Map2)
```



```
Map3 {dom=classdomainname} = iff(Map1="Forest",Map1,Map2)
```


Let us first look at a real example of a Map Calculation expression.

The price of the land will be 75 percent of the average value in the high mountain areas, above 3500 meters. The average landvalue is stored in the map `Landval`, that you have made earlier on.

☞

- Type the following expression on the command line:
  `Landval1{dom=value; vr=0:1000:1}= iff(Dem>3500,Landval*0.75,Landval);` ↵

Note that you have to enter a semicolon (;) after the Map Calculation expression. By entering the semicolon, the Raster Map Definition dialog box is skipped. As Map Calculation statements were already treated extensively in chapter 8, no more examples are given here.

For calculations in tables, you need to enter the word `Tabcalc` and the table name in front of a table calculation expression.

Some of the aspects of the use of expressions on the command line of the Main window, used for calculating with tables, are demonstrated with the same data set and analyses as treated in chapter 5: the table related to the city blocks of Cochabamba.

☞

- Open table `Citybl` and look at the meaning of the various columns.
- After that, close the table.

One of the things that can be calculated from this table is the population density per city block. The area of each city block is in meters, so we have to convert it to km^2 .

☞

- Type the following expression on the command line:
  `Tabcalc Citybl Popdens{dom=value; vr=0:1000:0.1}=1000* Population/Area;` ↵

- Open the table `Citybl` and look at the result.

As Table Calculation statements were already treated extensively in chapter 5, no more examples are given here. There are, however, some operations which are done via menus in the table window, that can also be executed by expressions: Namely *aggregations* and *table joining*. Some syntax examples are given in table 12.7.

Table 12.7: Overview of commands and expressions that can be used on the command line for aggregations in tables. In bold, the expression is shown that can be used on the command line of the table window. The entire part is used when executing it from the command line of the Main window, or in a script.

Aggregate function	**Expression**
Average	Tabcalc table **Outcol = ColumnAggregateAvg(col,group by,weight/1)** ;
Count	Tabcalc table **Outcol = ColumnAggregateCnt(col,group by,1)** ;
Maximum	Tabcalc table **Outcol = ColumnAggregateMax(col,group by,1)** ;
Median	Tabcalc table **Outcol = ColumnAggregateMax(col,group by,weight/1**) ;
Minimum	Tabcalc table **Outcol = ColumnAggregateMin(col,group by,1)** ;
Predominant	Tabcalc table **Outcol = ColumnAggregatePrd(col,group by,weight/1)** ;
Standard deviation	Tabcalc table **Outcol = ColumnAggregateStd(col,group by,weight/1)** ;
Sum	Tabcalc table **Outcol = ColumnAggregateSum(col,group by,1)** ;

Some of the aggregate functions allow the use of a weight column (average, median, predominant and standard deviation). The result can also be written into another table. In that case the expression becomes:

```
Tabcalc table.OutCcol=ColumnAggregateSum(Column,Group by column,1)
```

You can use this expression, for example to calculate the total area per district in the table `Citybl`.

☞

- Type the following expression on the command line:
  `Tabcalc Citybl Areadistrict {dom=value; vr=0:1000000000:1}`
  `= ColumnAggregateSum(Area,District,1);` ↵.

The process of table joining was extensively treated in exercise 5.5. In that exercise you have seen that there are different ways of joining tables, depending on whether you join via the domain of the table, or via the domain of a column, and on whether there is a one-to-one or a one-to-many relation between the items in the two columns with the same domain.

The syntax for the expressions that are used to perform table joining in the table calculation window are as follows:

- **Situation 1**: Joining of two tables with the same domain:

  ```
  OutCol = ColumnJoin(table.ext,column to join)
  ```

- **Situation 2**: Joining two tables, in which the domain of the current table (the table to which you join) is the same as the domain of a column of the second table (from which you join)

  A. There is a *one-to-one relation*: The domain items in the column you use to join from in the second table, occur only once.

  ```
  OutCol = ColumnJoin2ndKey(table.ext,column to join,key in second
  table)
  ```

  B. There is a *one-to-many relation*: The domain items in the column you use to join from, in the second table, occur more than once. In order to solve this you have to use an aggregate function.

  ```
  OutCol = ColumnJoinAvg(table.ext,column to aggregate,group by,
  weight)
  ```

- **Situation 3**: Joining two tables, in which the domain of the column in the current table (the table to which you join) is the same as the domain of the second table (from which you join).

  A. There is a *one-to-one relation*: The domain items of the column in the current table you use to join to occur only once and the domain items in the second table you use to join from are always unique (by definition).

  ```
  OutCol = ColumnJoin(table.ext,column to join,key column)
  ```

  B. There is a *many-to-one relation*: The domain items in the column of the current table that you use to join to are not unique, and the domain of the second table you use to join from are unique (by definition). In this case the values from the column to be joined will be repeatedly occurring after the joining.

  ```
  OutCol = ColumnJoin(table.ext,column to join,key column)
  ```

- **Situation 4**: Joining two tables, in which the domain of the column of the current table is the same as the domain of a column of the second table.

  A. There is a *one-to-one relation*.

  ```
  OutCol = ColumnJoin2ndKey(table.ext,column to join,key in first
  table,key in second table)
  ```

  B. There is *one-to-many relation*: The domain items in the column you use to join to, in the current table, occur only once, and those in the column of the second table occur more than once. In order to solve this you have to use an aggregate function.

```
OutCol = ColumnJoinAvg(table.ext,Column to be joined,Key in table
1,weight,group by)
```

C. There is *many-to-one relation*: The domain items in the column you use to join to, in the current table, occur more than once, and those in the column of the second table occur only once. In this case the values from the column to be joined will be repeatedly occurring after the joining.

```
OutCol = ColumnJoin2ndKey(table.ext,column to join,key in first
table,key in second table)
```

D. There is *many-to-many relation*: The domain items in the column you use to join to, in the current table, occur more than once, and those in the column of the second table occur more than once. In this case an aggregate function will be used that will result in repetitive values in the first table.

```
OutCol = ColumnJoinAvg(table.ext,Column to be joined,Key in table
1,weight,group by)
```

Whenever aggregations functions are used in table joining they will be similar to the ones shown in table 12.7.

Let's look at some examples of the use of expression for table joining. First an example of situation 1 is given: Joining of two tables with the same domain. In this case the attribute table `Citybl` and the polygon histogram `Citybl`.

☞

- Type the following expression on the command line:
  `Tabcalc Citybl Perimeter = ColumnJoin(Citybl.hsa, Perimeter); ↵`
- Open table `Citybl` and check the result.
- Close the table window.

The following example deals with situation 2B: The domain of the first table is the same as the domain of a column of the second table. There is a one-to-many relation. For example if we want to calculate the total population per district.

☞

- Type the following expression on the command line:
  `Tabcalc District Population = ColumnJoinSum(Citybl.tbt,Population,District,1) ; ↵`

12.1.7 Commands dealing with the management of dependencies

An important group of expressions deals with the management of *dependency links* between objects. In chapter 2, the concept of dependency links was explained extensively. When maps are used to create other maps, for instance by performing an operation or executing an expression, then this operation or expression and the input map name(s) are stored inside the new map. Output maps thus know how they are created and on which input maps, tables or columns they depend. Such output maps are called *dependent data objects*. The same applies for tables and columns.

The dependency concepts allow for the creation of a series of definition files first, before actually calculating maps, tables or columns. The calculation of the actual data files can be forced with the Calc command. The reverse, deleting the data files, and keeping the definition files (called *release disk space*), can be done with the Reldisksp command. When a source object is edited, the dependent objects can be updated, with the command update. table 12.8 gives a number of examples.

Table 12.8: Some examples of commands and expressions that can be used on the command line for management of dependency links

Command	**Example**	**Description**
domclasstoid	domclasstoid Geom	Change the object from class to Id domain.
domidtoclass	domidtoclass Citybl	Change the object from Id to class domain.
breakdep object.ext	breakdep Landuse.mpr	Break the dependency link of an object.
reldisksp object.ext	reldisksp Landuse.mpr	Release disk space of a dependent data object, by deleting the data file. Only the definition file remains.
update object.ext	update Slope.mpr	Make dependent maps and dependent tables up to date.
uptodatecol table.ext.col	uptodatecol Geom.tbt density	Makes a dependent column up to date.
calc object.ext	calc Slope.mpr	Calculates a map or table after it was defined but not yet calculated.
calccol table.ext.col	calccol Geom.tbt density	Calculates a dependent column after it was defined but not yet calculated.
delcol table.ext.col	delcol Geom.tbt density	Deletes a column in a table.

You will now practice with some of these expressions dealing with the management of dependencies. In the previous exercise you have created some dependent data objects which can now be used to demonstrate the expressions.

☞

- Type the following expression on the command line:
  `Reldisksp Landval.mpr ↵`
- The data of the raster map `Landval` will now be deleted. Only the definition file remains.

- Check this by opening the Properties dialog box of the map Landval. Then close it by pressing Cancel.
- Type the following expression on the command line:

  `calc Landval.mpr ↵`
- The raster map Landval will now be recalculated according to its definition.
- Check this by opening the Properties dialog box of the map Landval. Then close it.
- Type the following expression on the command line:

  `breakdep Landval.mpr ↵`
- The dependency link of the raster map Landval will now be broken. The map is no longer a dependent object, but a source object.
- Check this by opening the Properties dialog box of the map Landval.
- Type the following expression on the command line:

  `update *.* ↵`
- Now all maps in the directory c:\ilwis21\data\usrguide\chap12 will be updated.

Similar operations can be performed on dependent tables and column.

12.1.8 How to use the command line as a pocket calculator

Besides the many uses of the command line we have seen so far, there is still one which is worth explaining: The *pocket line calculator*. The Pocket Line Calculator is an ILWIS tool which enables the user to make quick calculations. All operators and functions described in the chapters Map Calculation and Table calculation are available. On the command line of the Main window type your calculation starting with a ?, which is in ILWIS the syntax for using the Pocket Line Calculator.

You may use the Pocket Line Calculator on the command line of a table window also. This enables you to retrieve quickly, some information on data in your table without creating new columns.

The pocket line calculator is especially convenient for:

– Quick calculations without the necessity to start other programs.

– Retrieving pixel information of maps (values, classes or ID’s, colors, coordinates etc.) without putting them on the screen.

– Trying out difficult calculations before performing them with maps or table columns.

Some examples of the use of the pocket line calculator are shown in table 12.9.

Table 12.9: Some examples of commands for using the pocket line calculator

Command	Example	Description
Arithmetic operations	? 7 + 8	Use it as a pocket calculator.
?Mapcolor(mapname, rowexpr, colexpr)	? mapcolor(Geom,100,200)	Returns the color of a pixel in a rastermap using the default representation.
Aggregation functions	?Avg(Area)	Returns the average value in the column area. Other aggregations are also possible: min, max, pred, std.
?mapmin(basemap)	?mapmin(Dem.mpr)	Returns minimum of raster, segment, polygon or pointmap basemap.
?mapmax(basemap)	?mapmax(Dem.mpr)	Returns maximum of raster, segment, polygon or pointmap basemap.
?pixsize(rasmap)	?pixsize(Dem)	Returns pixel size of raster map rasmap.mpr.
?pixarea(rasmap)	?pixarea(Dem)	Returns pixel area (square of pixel size) of georeference georef.grf.
?pixsize(georef)	?pixsize(Cochabam)	Returns pixel size of raster map rasmap.mpr.
?pixarea(georef)	?pixarea(Cochabam)	Returns pixel area (square of pixel size) of georeference goeref.grf.
?maprows(rasmap)	?maprows(Dem)	Returns nr. of rows of raster map rasmap.mpr.
?mapcols(rasmap)	?mapcols(Dem)	Returns nr. of columns of raster map rasmap.mpr.
?tblrecs(tbl)	?tblrecs(Cirtybl)	Returns nr. of records in table tbl.tbt.
?tblcols(tbl)	?tblcols(Citybl)	Returns nr. of columns in table tbl.tbt.

☞

- Type the following command on the command line:
  `?100/20 ↵`
- The result (5) is given in a small message box.
- Click OK.
- Type the following expression on the command line:
  `?pixsize(Dem) ↵`
- The result (20) is given in a small dialog box.
- Click OK.
- Close all map and table windows that are still open.

More complicated expressions or expressions which will be used several times may be stored as a *user defined function*. These will be treated in the next exercise.

Summary: Working from the command line

Most of the activities that are done via the use of menus, can also be done via the command line of the Main window, located at the top of the Main window, just below the menu bar.

The command line is used for the following activities:

– To perform calculations with maps and attribute tables. The calculations with tables were treated in chapter 5, and the Map Calculation statements were discussed in chapter 8.

- To perform ILWIS commands: To display, edit, create or view the properties of ILWIS objects, and to obtain dialog boxes to start an ILWIS operation. This functionality is equivalent to typing an ILWIS 1.4 executable name on the DOS command line in ILWIS version 1.4. You can also use ILWIS commands for data management: copying and deleting objects, breaking dependency links of objects.
- To perform ILWIS expressions: To perform complete ILWIS operations. This functionality is equivalent to typing the ILWIS 1.4 executable name and all parameters required by this executable on the DOS command line in ILWIS version 1.4.
- To run scripts, which usually contain a sequence of ILWIS expressions. With a script, you can build a complete GIS and Remote Sensing analysis/application for your own research discipline. Scripts are more or less equivalent to batch files in ILWIS version 1.4.
- To use it as a pocket calculator. If you put a question mark before an expression, the result of that expression is shown on the command line. For example: When you type ?100/5 ↵ , the command line will show the value 20.
- To start any Windows application program, batch file, or DOS application (with a .PIF file available), by writing an exclamation mark in front of the application name. Applications that can be started from the command line may have the following extensions: .exe, .com, .bat, .pif. Type the application name directly after the exclamation mark (no spaces allowed). For example, to start Word, type: !Winword.

12.2 Functions

In Chapter 5 you have seen the use of calculation formulas for working with tables, and in chapters 7 and 8 those for working with maps. As you have seen, there are many different operators and functions that can be applied on value maps and on class or ID maps. A complete overview of the operators and functions available in Table calculation and in Map Calculation, can be found in the On-Line Help, together with a series of examples.

In this exercise you will first have a look at some examples of functions that are already present in the system (pre-programmed functions), before you can practice with the creation of your own functions (user-defined functions).

System-defined functions

A number of these functions where already treated in chapters 5, 7 and 8. Here, only some examples of system-defined functions are given. One of the most important functions is the *Conditional IF function.*

IFF(*a*,*b*,*c*)	If condition a is true, then return the outcome of expression b, or else (when condition a is not true) return the outcome of expression c. Mind the double ff in iff (standing for IF Function).

☞

- Type the following command on the command line:
  `Result1=IFF(Dem>2400,10,0); ↵`

It means: If a pixel in map `Dem` (digital elevation model) has a value greater than 2400, then assign the value 10 to this pixel in output map `Result1`, or else assign a 0.

Random functions

For statistical purposes you might need a map with random values.

RND(long)	Returns random long integer values in the range [1; 2 billion ($2*10^9$)]; To simulate a die, use this function in the form of: RND(6).
RND(0)	Returns a 0 or 1 at random.
RND()	Returns random real values in the range [0;1> , i.e. between 0 and 1, including 0 but excluding 1.

For example, if you want to subdivide your map randomly into two groups of pixels, use the following formula:

☞

- Type the following command on the command line:

  `Map1 = iff( Dem = 1 , 1, 1)↵`
- The Raster Map Definition dialog box is opened. Select the domain value, and the value range 0 to 1, and the precision 1.0. Click OK.
- Type the following command on the command line:

  `Mapran:=RND(0)*Map1`

in which `Map1` is a georeferenced value map with value 1 for every pixel. `Map1` can be calculated from any map using the appropriate georeference. `Mapran` uses the same georeference as `Map1`. The pixels in the output map will randomly get the value 0 or 1. Random functions are very useful for all kinds of statistical testing.

☞

- The Raster Map Definition dialog box is opened. Select the domain `value`, and the value range `0` to `1`, and the precision `1.0`. Click OK.
- Open the map `Mapran` and examine the results. Close the map window.

MinMax functions

MIN(*a*,*b*)	Calculates the minimum of two expressions *a* and *b*.
MIN(*a*,*b*,*c*)	Calculates the minimum of three expressions *a*,*b* and *c*.
MAX(*a*,*b*)	Calculates the maximum of two expressions *a* and *b*.
MAX(*a*,*b*,*c*)	Calculates the maximum of three expressions *a*,*b* and *c*.

Using these functions, you can for instance calculate for each pixel the minimum or maximum value of 2 or 3 input maps; substitute *a*,*b*,*c* with the map names.

☞

- Type the following command on the command line:

  `Min3=min(Tmb1,Tmb2,Tmb3)↵`
- The Raster Map Definition dialog box is opened. Select the domain image. Click OK.
- Open the map `Min3` and examine the results. Close the map window.

User defined functions

Besides many internal pre-programmed functions, ILWIS gives the user an opportunity to create new functions. They may be used in all four calculators in ILWIS: MapCalc, TabCalc, Scripts or in the pocket line calculator. Especially when you need to execute certain calculations, which require a lot of typing work several times, user-defined functions may be time saving. A user-defined function is an expression which may contain any combination of operators, functions, maps and table columns. Firstly you will create a simple function and after that a more complex one.

> ☞
>
> - Double-click New Function in the Operation - list. The Create function dialog box is opened.
> - Type the Function Name: `Avgb`
> - Type the Expression: `(a + b)/2`
> - Type the Description: `averaging two value maps`. Click OK.

The Edit Function dialog box appears showing your newly created function. If necessary, you can edit your function in this dialog box. The different parameters in your function may be names of maps or table columns, or you use characters *(a,b)* which you may specify later when you apply the function. The function in the function editor is defined as follows:

1	`Function avgb(Value a,Value b) : Value`
2	`Begin`
3	`Return (a+b)/2;`
4	`End;`

The line numbers do not form part of the function. They are only used here to explain the contents.

– In line 1 the function name is given and the parameters, between brackets. In this case there are two parameters: Value *a* and Value *b*. Also the output domain is given: Value.

– In line 2 the word `Begin` indicates the beginning of the actual function expression, which is given in line 3. Note that the expression ends with a semicolon (;). In line 4 the end of the function is indicated with the word End.

Type your function on the command line of the Main window or table window. Start with an output map name (or column name) followed by the definition symbol (=), the name of your function and parameters. The parameters, replacing the characters *a, b, c*, etc. in your function, have to be entered in brackets separated by commas. Of course, the parameter which is filled in first, is taken as the first parameter encountered in your user-defined function.

☞

- Click OK in the Edit Function dialog box.
- Type the following expression on the command line:
  `Avgtm=avgb(Tmb1,Tmb2)`↵
- The Raster Map Definition dialog box is opened. Select the domain `image`. Click OK.
- Open the map `Avgtm` and examine the results.
- Close the map window.

The following example shows a more complex expression. We will calculate the direction of slopes and create a so called *aspect map* (see chapter 10). An aspect map (slope direction map) is calculated using the formula:

```
Aspect = RADDEG( ATAN2 (Dx / Dy)+ pi)
```

When you want to use this formula more often it is convenient to put the formula in a function. You can create the function aspect which has two variables for `Dx` and `Dy`. Later, when we apply the function, you substitute dx and dy with the real names of the maps for the horizontal and vertical gradient.

☞

- Double-click New Function in the Operation-list. The Create function dialog box is opened.
- Type for the Function Name: `Aspect`
- Type for the Expression: `RADDEG( ATAN2 (Dx , Dy) + pi)`
- Type the Description: `slope aspect between 0 and 360 degrees`.
- Click OK. The Edit Function dialog box is opened.
- Click OK.

As you can see ILWIS assumes that PI is another variable (a map or value). But in fact PI represents here the system-defined variable. So you should edit the function to remove the variable PI.

The correct definition of the function should be:

```
Function aspect(Value dx,Value dy) : Value
Begin
  Return Raddeg(ATAN2(dx,dy)+PI);
End;
```

☞

- Edit the Function until it is the same as above.
- Click OK.

To apply the function `Aspect`, in the command line of your Main window:

☞

- Type the following command on the command line:
  `Aspect = aspect(dx,dy)↵`
- The Raster Map Definition dialog box is opened. Select the domain `value`, and the value range from `0` to `360`, with a precision of `1`. Click OK.
- Open the map `Aspect` and examine the results. Close the map window.

Now you can easily calculate several aspect maps from for instance other areas. You only have to define your new input variables for the function. Which means you give the new `Dx` and `Dy` map from another area.

Summary: Functions

– ILWIS contains over 50 different functions that are pre-programmed and that can be used in Table calculation, in Map Calculation or in the pocket line calculator.

– Besides many internal pre-programmed functions, ILWIS gives the user an opportunity to create new functions.

– A user-defined function is an expression which may contain any combination of operators, functions, maps and table columns.

– They may be used in all four calculators in ILWIS: MapCalc, TabCalc, Scripts or in the pocket line calculator.

– User-defined functions are especially useful when you need to execute certain calculations which require a lot of type work.

– User-defined functions can be created with the New Function operation.

– Type your function on the command line of the Main window or table window. Start with an output map name (or column name) followed by the definition symbol (=), the name of your function and parameters. The parameters, replacing the characters *a*, *b*, *c*, etc. in your function, have to be entered in brackets separated by commas. Of course, the parameter filled in first, is taken as the first parameter encountered in your user-defined function.

12.3 Scripts

In the previous exercises you have seen two important tools for automating analysis procedures in ILWIS: The use of commands and expressions on the command line, and the use of user-defined functions. The third and most important tool is *scripts.* A script is a list of commands and expressions. With the help of a script, a complete GIS or Remote Sensing analysis can be performed automatically. A script may contain all the commands and expressions that were treated in exercise 12.1. These include commands and expressions for the creation and calculation of data objects, for object management (e.g. copy or delete), and for display of data objects (Open and Show). Other scripts and other Windows applications can also be called from within a script.

A script name must start with a character between A-Z and cannot exceed 8 characters. A script consists of an object definition file with the extension `.isl` (ILWIS Script Language) and a data file with the extension `.isf` (ILWIS Script File). A script can be started from the command line of the Main window by typing:

```
Run scriptname ↵
```

Creating a Script

In this exercise a script is created to calculate two slope maps: One representing slope steepness in percentages and one in degrees. The segment map `Contour`, and ILWIS standard filters `DFDX` and `DFDY`, are used to calculate the slope maps for the Cochabamba area. The procedure for creating slope maps is treated extensively in chapter 10.

- Double-click the icon NewScript in the Operation - list.

The Create Script dialog box is opened. The script that will be needed to calculate a slope map, starting from a contour map, is shown below. The numbers on the left side are not part of the script. They are used here as a reference for explaining the meaning of the various lines.

	Rem ILWIS script for calculating slope maps
1	`dem = MapInterpolContour(contour,cochabam)`
2	`dx = MapFilter(dem, dfdx)`
3	`dy = MapFilter(dem, dfdy)`
4	`slopeper = ((HYP(dx,dy)) / pixsize(dem)) * 100`
5	`slopedeg = RADDEG (ATAN ((HYP(dx, dy)) / pixsize(dem)))`
6	`calc slope*.*`
7	`open slopedeg`

For a better understanding of the script statements is recommended to repeat the exercises in chapter10. Here only a brief explanation is given.

- Line 1: To use operation InterpolContour to create an interpolated height map from segment contour lines. The expression is :
  `dem = MapInterpolContour(contour, cochabam).` Perform a contour interpolation on segment map `Contour`, use georeference `'Cochabam'`, and write the output to map `Dem`.
- Line 2: To use filter `Dfdx` on the interpolated contour map to calculate height differences in X-direction. The expression is:
  `dx= MapFilter(dem, dfdx).` Filter map `Dem` with the `Dfdx` filter and write the output to map `Dx`.
- Line 3: To use filter `Dfdy` on the interpolated contour map to calculate height differences in Y-direction The expression is:
  `dy = MapFilter(dem, dfdy).` Filter map `Dem` with the `Dfdx` filter and write the output to map `Dy`.
- Line 4: To calculate a slope map from `Dx` and `Dy`, the Map Calculation expression is:
  `slopeper = ((HYP(dx,dy)) / pixsize(dem)) * 100.`
  HYP is an internal MapCalc/TabCalc function; `Dx` and `Dy` are the output maps from the filtering; `pixsize(dem)` calculates the pixel size of the map `Dem`, `Slopeper` is the output map name of the map containing slope value in percentages.
- Line 5: To convert the percentage values into degrees, another Map Calculation expression is used:
  `slopedeg = RADDEG(ATAN((HYP(dx,dy))/pixsize(dem)))`
  Funcions `ATAN`, `HYP` and `RADDEG` are internal MapCalc/TabCalc functions.

☞

- Type `Slopecal` in the text box Script Name.
- Type the following lines in the text box for script definition:

```
//script to calculate slope maps in percentage and
in degrees.
//Dem=MapInterpolContour(Contour,Cochabam)
Dx=MapFilter(Dem,Dfdx)
Dy=MapFilter(Dem,Dfdy)
Slopeper=((HYP(dx,dy))/pixsize(dem))*100
Slopedeg=RADDEG(ATAN((HYP(dx,dy))/pixsize(dem)))
calc slope*.*
open slopedeg.mpr
```

- Click OK.

The script `Slopecal` is created. Note that the first two lines start with //. This indicates that the lines contain a remark, and will not be executed by ILWIS. We have put a // before the actual contour interpolation, since this takes quite some time. To run the script:

☞

- Type on the command line in the Main window:

  `run Slopecal ↵`

The result of running this script are the maps `Slopeper` and `Slopedeg` which are slope maps in percentages and in degrees.

☞

- Check the contents of the maps `Slopeper` and `Slopedeg`. Close the map windows.

How do you know the correct script syntax

The exact syntax for the script statements is not something you know by heart, or maybe you can learn it after working a considerable time with ILWIS. There are, however, several methods which you can use to create expressions for scripts:

– Use the menu and the dialog boxes for a certain operation. Fill in all required parameters in the dialog box, and click OK. At that moment the expression for that operation is shown on the command line. You can then copy the resulting expression from the command line into the script. Within the script editor dialog box and the command line, you can use the following key strokes:

  CTRL+C Copy the selected part to the clipboard.

  CTRL+V Paste the contents from the clipboard.

– The *ILWIS log file*. ILWIS keeps track of everything you are doing in a so-called *log file*. The ILWIS log file is called *Ilwis.log*, and it can be found in the start-up directory that you have specified for the ILWIS program. This will normally be the directory c:\ilwis21\data, unless you have changed it yourself, by changing the Program Item Properties of the ILWIS program in the Windows Program Manager. The log file is an ASCII file that you can open with a text editor. You can copy part of the expressions that are stated in the log file to a script.

On the command line, script parameters are entered without any brackets, separated by spaces. If a script does not contain parameters, you can directly double-click the script icon in the Catalog. Parameters in a script have to be filled out when the script is run by the program. In the following sections some examples of scripts are given.

Example of a script for table calculation

In section 5.6 you have been practicing with table joining for an urban problem, using two tables: `Citybl` (table linked to the map `Citybl`, with information on the 717 city blocks in the central part of Cochabamba), and `District` (a table with information on the cadastral districts of the city). In the last part of the exercise you solved the following problem:

Calculate the total area and the total population for each district. Apart from that, calculate the percentage cover of residential, commercial and institutional buildings in each district. Another thing we want to know is the relation between the number of schools and the number of schoolchildren (under 18 years old) for the districts of Cochabamba city. In order to solve this problem, we need to know the land use types, the area, the population, the number of school children, and the number of schools in each district.

The information on areas, land use types and population is available for each city block, in the table `Citybl`. The information on the number of schools and the percentage of schoolchildren of the population is known per district, stored in the table `District`. Since you know for each city block in which district it is located, you can use the information from the table `Citybl` and bring it into the table `District`. However, the table `Citybl` contains 717 records and the table `District` only 13. So you will have to do an aggregation.

The script for calculating this problem is shown below:

```
   rem ILWIS Script
1  opentbl citybl.tbt
2  Tabcalc citybl areadistrict=ColumnAggregateSum(Area,District,1)
3  Tabcalc citybl distrlanduse= district + landuse
4  Tabcalc citybl areadistrlu = ColumnAggregateSum(Area,Distrlanduse,1)
5  Tabcalc citybl residential {dom=perc; : :1} = iff(landuse="residential",100*areadistrlu/areadistrict,0)
6  Tabcalc citybl commercial {dom=perc; : : 1} = iff(landuse="commercial",100*areadistrlu/areadistrict,0)
7  Tabcalc citybl institutional {dom=perc; : :1} = iff(landuse="institutional",100*areadistrlu/areadistrict,0)
8  closetbl citybl.tbt
9  rem open the table district
10 Opentbl district.tbt
11 Tabcalc district residential = ColumnJoinMax(citybl.tbt,residential,District,1)
12 Tabcalc district commercial = ColumnJoinMax(citybl.tbt,commercial,District,1)
13 Tabcalc district institutional = ColumnJoinMax(citybl.tbt,institutional,District,1)
14 Tabcalc district Population = ColumnJoinSum(citybl.tbt,Population,District,1)
15 Tabcalc district children=population*pchildren/100
16 Tabcalc district childpschool=children/schools
17 open district.tbt
```

The line numbers in the table do not form part of the script. They are only used here to comment on the various expressions.

– In line 2 the aggregate function `Sum` is used to sum up the area per district. The result is stored in the column `Areadistrict` of the table `Citybl`.

- In line 3 (`DistrLanduse= district + Landuse`) the two columns `Landuse` and `District` are combined (concatenated) into a new column.
- In line 4 the aggregate function Sum is used to sum up the area of each landuse type per district.
- In lines 5, 6 and 7 the percentage coverage of residential, commercial and institutional areas within each district is calculated. Note that the domain of the output column is specified: `Perc`, which is the percentage domain, with a precision of 1.
- In line 8 the `Citybl` table is closed, and in line 10 the `District` table opened.
- In lines 11, 12 and 13 the percentage cover values for residential, commercial and institutional areas within each district, stored in table `Citybl`, are read into the table `District`. The domain of the table `District` is joined with the domain of the column `District` in the table `Citybl`. The aggregate function is needed, since 1 record of a district from the table `District`, is linked to many records of the same `District` in table `Citybl`.
- In line 14 another join operation is performed to read in the population data from the table `Citybl`, summed up for each district, into table `District`.
- In line 15 this total population per district is used in combination with the percentage of schoolchildren per district, to find the number of schoolchildren per district.
- In line 16, the number of children per school is calculated for each district, and the table is finally displayed in line 17.

☞

- Click the Customize Catalog button in the button bar of the Main window and select all object types. Click OK.
- Click with the right mouse button on the script `Urban` and select Open from the Context-sensitive menu. The Edit Script urban dialog box is opened, in which you will see the script statements.
- Close the script editor by clicking the Cancel button.
- You can run the script by typing the following expression on the command line:

  `run Urban ↵`
- Have a look at the result in the table `District`. Close the table after you have seen the result.

Example of a script for Map Calculation

In chapter 8 you have been working with Map Calculation formulas in the analysis of a simple, hypothetical problem, dealing with the calculation of the price of the land in the Cochabamba region.

Now you will do the same analysis, using a script.

The average land prices per hectare are given in an attribute table linked to the land use map. However, these average values will either be higher or lower, depending on a set of criteria:

- 1. The price of the land will be 100 percent of the average value when located on slopes of less than 20 degrees, and 70 percent when located on slopes of more than 20 degrees. Slope information is stored in the map `Slope`.
- 2. The price of the land will be 40 percent of the average value when it is located on an active landslide or on an area with high erosion, and 60 percent when located on an old landslide. For this criterion we need the geomorphologic map (`Geom`)

When evaluating the combination of criteria we only look at which of the criteria will lead to the lowest land price.

Please keep in mind that the objective of this exercise is not that you learn about an application - for that the problem is too hypothetical - but that you learn how to use Map Calculation formulas. The Application handbook focuses more on the applications than on the tools. The script looks as follows:

	rem ILWIS Script
1	geom.mpr{dom=geom} = MapRasterizePolygon(geom,cochabam.grf)
2	landuse.mpr{dom=landuse} = MapRasterizePolygon(landuse,cochabam.grf)
3	landval = (landuse.landvalue) / 25
4	landval1 = iff(slope > 20 , landval * 0.7 , landval)
5	landval2 = iff(geom="ol", landval * 0.6, iff((geom = "al") or (geom = "he"), landval * 0.4 , landval))
6	landvalc = min(landval1,landval2)
7	landvalf = iff(isundef(geom) , landval2, landvalc)

The line numbers in the table do not form part of the script. They are only used here to comment on the various expressions.

For a better understanding of the script statements it is recommended to repeat the exercise in section 8.1. Below, only a brief explanation on the script lines is given.

- In lines 1 and 2 the maps `Geom` and `Landuse` are rasterized, using the georeference `Cochabam`.
- In line 3 the map `Landuse` is renumbered, with the values from the column `Landvalue` in the attribute table linked to the map `Landuse`. The land use map is linked to an attribute table, in which the average land value (per hectare) is stored for each land use type. Since the average land values are given per hectare, and you are working on maps with a pixel size of 20 meters,

you need to divide the land value by 25 in order to obtain the average value per pixel.

– In line 4 the first criterion is applied: If the slope is more than 20 degrees, then the price of the land will only be 70 percent of the average value.

– In line 5 the second criterion is applied: If the pixel is an old landslide, then the value is only 60 percent of the average value. If the pixel is on an active landslide or on an active erosion area, the value is only 40 percent of the average. The information on landslides and erosion is stored in the map `Geom`. Codes are used instead of the names of the geomorphologic units. The unit "`old landslide`" in the domain `Geom` has the code "`ol`". If you use codes, the formulas can be much smaller.

– Now you have generated two maps that contain land values based on one criterion (`Landval1` and `Landval2`). What should you do for pixels where more than one of these criteria occurs? For example: Pixels with a slope less than 20 degrees, located on an active landslide. The best here is to take for each pixel the minimum of the same pixel in one of the two maps. This is done in line 6.

– Since the map `Landvalc` occupies a smaller area than the map `Landval` the formulas will result in undefined values, for those places where one of the input maps is undefined. This is corrected in line 7.

☞

- Click with the right mouse button on script `Landval` and select Open from the Context-sensitive menu. The Edit Script landval dialog box is opened, in which you will see the script statements.
- Close the script editor by clicking the Cancel button.
- You can run the script by typing the following expression on the command line:

  `run Landval ↵`

Since all the expressions in the scrip landval are written with the definition symbol (=) only the definitions of the maps are stored. The maps are not calculated until you open them. When you open the last map (`Landvalf`) all previous maps are also calculated.

☞

- Open the map `Landvalf`. The calculation starts with the first map which was defined in the script. The total calculation takes a few minutes. Have a look at the result and close the map window.

Using parameters in a script

In the previous examples you have only looked at scripts that are made for specific maps or tables. You can also make scripts that are more widely applicable, by introducing parameters. A map, table or column can be represented by parameters in a script. Parameters in a script must be written as %1, %2, %3, etc. (access On-line Help or Reference Guide for more information).

In this exercise you will calculate the density of landslides in the units of various maps (the landuse map, the geological map, the classified slope map). Before you can run the script that does the actual density calculation, you first need to create a map indicating the landslides. This will also be done with a small script. The script will look like this:

	rem ILWIS Script: for creating a map with landslides
1	crdom slide -type=class
2	additemtodomain slide "landslide"
3	additemtodomain slide "no landslide"
4	slide {dom=slide}=iff((geom="al")or(geom="ol"),"landslide","no landslide")
5	calc slide.mpr

The line numbers in the table are not forming part of the script. They are only used here to comment on the various expressions. The script contains the following expressions:

- In line 1 a class domain is created with the name `Slide`.
- In lines 2 and 3 two items are added to this domain.
- In line 4 the raster map `Slide` is defined which will have one of the two items, defined in line 2 and 3. The Map Calculation formula uses as input the geomorphologic map `Geom`. The codes “`al`” and “`ol`” stand for “`active landslide`” and “`old landslide`”.
- In line 5 the map `Slide` is calculated.

☞

- Open the script `Slide` and look at its contents.
- Close the script editor by clicking the Cancel button.
- You can run the script by typing the following expression on the command line:

  `run Slide ↵`
- Have a look at the result map `slide`.
- Close the map window after you have seen the result.

Now that the input map slide is finished you can do the actual density calculation, using the script `Density`. This scripts contains the following expressions:

```
  rem ILWIS Script for calculating landslide density for different maps
1 s%1 = TableCross(%1,slide)
2 calc s%1.tbt
3 Tabcalc %1 areaclass {dom=value; vr= 0 : 100000000000 : 1} = ColumnJoinSum (s%1.tbt,Area,%1,1)
4 Tabcalc s%1 areasl {dom=value; vr= 0 : 100000000000 : 1 } = iff(slide="landslide", area , 0)
5 Tabcalc %1  areaslide {dom=value; vr=0 : 100000000000 : 1} = ColumnJoinSum (s%1.tbt,areasl,%1,1)
6 Tabcalc %1 density {dom=perc} = 100* areaslide/areaclass
7 calc s%1.tbt
8 calc %1.tbt
```

The line numbers in the table are not forming part of the script. They are only used here to comment on the various expressions. The script contains the following expressions:

- In line 1 a raster map, indicated with parameter %1, is crossed with the map `Slide`. The result is stored in a cross table, named `s`+the name of the input raster map. So if we use the map `Landuse` for example, the output cross table is called `Slanduse`. In this line only the dependent cross table s%1 is defined, but not yet calculated.
- In line 2 the cross table is actually calculated.
- In line 3 the total area of each domain item in the raster map %1 is defined, using the aggregate function SUM on the column `Area` in the cross table. The aggregation is done after grouping the data in the table according to the domain items %1. Note that the result of the formula is not written in the cross table s%1, but in the attribute table linked to the map.
- In line 4 a new column `Areasl` is defined, in which the records in the cross table that have a combination between a domain item of the raster map and the unit “`landslide`” in the map slide, are assigned the value of the column `Area` . The combination of the domain items and the unit “`no landslide`” will receive a value 0. This is done so that in the next line we can know the area of landslides within each domain item. Note that the result of this formula is written in the cross table.
- In line 5 the area of each domain item in the raster map %1 that is occupied by landslides is defined, using the aggregate function SUM on the column `Areasl` in the cross table. The aggregation is done after grouping the data in the table according to the domain items %1. Note that the result of the formula is not written in the cross table s%1, but in the attribute table linked to the map.
- In line 6 the density of landslides is calculated by dividing in the attribute table %1, the area occupied by landslides in each class, by the total area of the class,

and multiplying the result by 100. The result will be percentages. That is why the domain Perc is selected for the output column.

– In line 7 the cross table is calculated so that the expression in line 5 is actually stored as values.

– In line 8 the attribute table is calculated so that the expressions in line 3, 5 and 6 are stored as values.

Note that this script only contains one parameter: %1, which is the name of the map that is used for the calculation of densities.

When you run a script with parameters these should be defined after the name of the script. So in this case: `run density Landuse`. ILWIS will then replace every time the parameter definition %1 by the word `Landuse`. The script will be interpreted as follows:

```
  rem ILWIS Script for calculating landslide density for different maps
1 slanduse = TableCross(landuse,slide)
2 calc slanduse.tbt
3 Tabcalc landuse areaclass {dom=value; vr= 0 : 100000000000 : 1} = ColumnJoinSum (slanduse.tbt,Area,landuse,1)
4 Tabcalc slanduse areasl {dom=value; vr= 0 : 100000000000 : 1 } = iff(slide="landslide", area , 0)
5 Tabcalc landuse  areaslide {dom=value; vr=0 : 100000000000 : 1} = ColumnJoinSum (slanduse.tbt,areasl,landuse,1)
6 Tabcalc landuse density {dom=perc} = 100* areaslide/areaclass
7 calc slanduse.tbt
8 calc landuse.tbt
```

☞

- Open the script `Density` and look at its contents.
- Close the script editor by clicking the Cancel button.
- You can run the script by typing the following expression on the command line:

  `run density Landuse ↵`
- Have a look at the resulting attribute table `Landuse`. It will contain three new columns: `Areaclass`, `Areaslide` and `Density`. Close the table after you have seen the result.

The same script can now be used for calculating the landslide density in other maps, such as `Geology`, `Slopecl`, or `Catchme`.

Running a script from another script

It is also possible to run one script from another script by including the expression: `Run scriptname parameter`. In this example we could make another script, that serves as the input for the script density, that you have just seen. The script density could then be calculated for different maps. The script that serves as input would look like:

	rem ILWIS Script for the input for the script density
1	run density geology
2	run density slopecl
3	run density catchmen

The line numbers in the table are not forming part of the script. They are only used here to comment on the various expressions. The script contains the following expressions:

– In line 1 the script `Density` is executed with the map `Geology` as parameter.

– In line 2 the script `Density` is executed with the map `Slopecl` as parameter. Etc.

☞

- Create a script `Densin` and enter the lines as given in the example (without the line numbers).
- Close the script editor by clicking the OK button.
- You can run the script by typing the following expression on the command line:

  `run Densin ↵`
- Have a look at the resulting attribute tables `Geology, Slopecl and Catchme`.
- Close the tables after you have seen the result.

Summary: Scripts

– Scripts are used to automate the operations in ILWIS.

– A script is a list of commands and expressions.

– With the help of a script, a complete GIS or Remote Sensing analysis can be performed automatically.

– A script may contain all the commands and expressions that were treated in exercise 12.1. These include commands and expressions for the creation and calculation of data objects, for object management (e.g. copy or delete), and for display of data objects (Open and Show).

- A script name must start with a character between A-Z and cannot exceed 8 characters. A script consists of an object definition file with the extension `.isl` (ILWIS Script Language) and a data file with the extension `.isf` (ILWIS Script File).
- A script can be made by copying the expression on the command line after you have filled in all required parameters in the dialog box of a certain operation, and clicked OK. At that moment the expression for that operation is shown on the command line. You can then copy the resulting expression from the command line into the script.
- A script can also be made by copying parts from the *ILWIS log* file after you have executed an operation via dialog boxes. ILWIS keeps track of everything you are doing in a so-called log file. The ILWIS log file is called *Ilwis.log*, and it can be found in the data directory that you have specified for the ILWIS program. It can be opened with a text editor.
- A script can be started from the command line of the Main window by typing: `Run scriptname ↵`
- A map, table or column can be represented by parameters in a script. Parameters in a script must be written as %1, %2, %3, etc.
- Other scripts and other Windows applications can also be called from within a script.