CHAPTER 12

Scripts and functions

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.

Spatial data analysis can also be performed in ILWIS by typing *commands*, *calculations* or *expressions* on the Command line of the Main window, or by combining these in a *script*.

When using a script, a series of commands, calculations and expressions are predefined in a list. These are consecutively executed when the script is run. Furthermore, a script may contain variables, so that the script can be used for different input maps, tables or values. Scripts can save a lot of time when dealing with frequently occurring analyses.

The script language is similar to the Command line syntax and we will therefore have a closer look at the Command line in the first part of this chapter. We will then start creating scripts with and without variables.

In chapter 5 (attribute data handling) and in chapters 7, 8 and 9, you have been working with the Command line to perform calculations on tables and maps. Instead of typing long and similar calculations on the Command line for different input maps one by one, you can store frequently used calculations as *functions*, which can then be used for different maps. The use of functions will be treated in the last part of this chapter.

Before you can start with the exercises, you should start up ILWIS and change the subdirectory to C:\ILWIS 3.0 Data\Users Guide\Chapter12 or to the directory where the data files for this chapter are stored.

☞

- Double-click the ILWIS icon on the desktop.
- Use the Navigator to go the directory: C:\ILWIS 3.0 Data\Users Guide\Chapter12.

12.1 Creating a script

In this part you will learn how to create a script. 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 are listed in the ILWIS Help topic Appendices, ILWIS script language (syntax). 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.

The script language is similar to the Command line syntax and we will therefore have a closer look at the Command line first.

12.1.1 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 toolbars.

The Command line can be used for the following activities:
- To perform *ILWIS commands*: To display, edit or create ILWIS objects, and to obtain dialog boxes to start an ILWIS operation. You can also use some special script language for *data management*: copying and deleting objects, and breaking dependency links of objects;
- 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 expressions* allowing you to perform complete ILWIS operations directly from the Command line;
- To generate *script* syntaxes, that can be copied to a script.

! The Command line has a history: use the List button ▾ or the Arrow up key to retrieve previously used expressions and commands. The Escape key can be used to clear the Command line.
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.

To perform an ILWIS command, you just type it on the Command line. Some examples of commands are listed in Table 12.1. Since the Command line is case insensitive, it doesn’t matter if you type them in upper or lowercase. A complete overview of ILWIS commands can be found in the ILWIS Help topic, Appendices ILWIS script language (syntax).

☞

- Open Help, go to the Appendices and browse through the following topics: ILWIS commands and expressions: ILWIS commands, ILWIS expressions, ILWIS expressions (alphabetic), Construction of expressions, ILWIS script language (syntax).

Table 12.1: Some commands that can be used on the Command line.

Command	Example	Description
Open `object.ext`	Open `Geomorphology.mpr`	Opens the Display Options dialog box for the object or shows the object immediately (e.g. tables)
Edit `object.ext`	Edit `Geomorphology.mpa`	Opens polygon map `Geomorphology` in the Polygon editor
Copy `object.ext objname`	Copy `Cityblocks.mpa Cocha`	Copies polygon map `cityblocks` to a new name
Del `object.ext`	Del `temp.mpr`	Deletes raster map Temp
Md `directory`	Md `temp`	Creates a new directory `Temp`
Help	Help	Opens the ILWIS Help

☞

- Type the following command on the Command line: `Open` ↵
- The Open Object dialog box appears from which you can select any object.
- Select raster map `Geomorphology` and click OK. The Display Options - Raster Map dialog box is opened. Click OK. The raster map `Geomorphology` is displayed.
- Close the map window.
- Type the following command on the Command line:
  `Open Geomorphology` ↵

You will notice that you get an error message when you type: `Open Geomorphology` ↵.
This is due to the fact that ILWIS doesn't know which object with the name `Geomorphology` should be displayed: a raster map, a vector map, a table, a domain, etc. You have to use the name of the object and its extension.

☞

- Type the following command on the Command line:
  `Open Geomorphology.mpr` ↵
- Click OK in the Display Options – Raster Map dialog box.
- Close the map window.

The extension `.mpr` is the extension of raster maps. In Table 12.2, the extensions of some ILWIS objects are listed.

Table 12.2: File extensions of different ILWIS objects.

Extension	Example	Description
.mpr	`Geomorphology.mpr`	File extension of raster maps.
.mpa	`Cityblock.mpa`	File extension of polygon maps.
.mps	`Contour.mps`	File extension of segment maps.
.mpp	`Rainfall.mpp`	File extension of point maps.
.tbt	`Cityblock.tbt`	File extension of tables.
.mpv	`Cochabamba.mpv`	File extension of map view.
.mpl	`Tms.mpl`	File extension of a map list.
.ioc	`Bolivia.ioc`	File extension of an object collection.
.ilo	`Cochabamba.ilo`	File extension of a layout.
.his	`Slope.his`	File extension of a raster histogram.
.hsa	`Landuse.hsa`	File extension of a polygon histogram.
.hss	`Drainage.hss`	File extension of a segment histogram.
.hsp	`Wells.hsp`	File extension of a point histogram.
.rpr	`Landuse.rpr`	File extension of a representation.

To practice with some commands, expressions and script language, we will now calculate a landslide risk map. When a unit in the `Geomorphology` map is classified as active landslide (AL) or old landslide (OL), we will classify it as landslide in the new map. The new map `Slide` will contain 2 classes, `Landslide` and `No Landslide`. We will therefore start with the creation of a new domain containing these 2 classes.

In the next part of this chapter we will create the same landslide map, but then using a script.

☞

- Type the following command on the Command line: `Crdom Slide` ↵
- The class domain `Slide` is created, but it still doesn't contain any items.
- Type the following command on the Command line:
  `Additemtodomain Slide “Landslide”` ↵
  `Additemtodomain Slide “No Landslide”` ↵
- The domain `Slide` now has two items. Check this by opening the domain.
- Enter the following MapCalc statement:
  `Slide{dom=Slide}=IFF((Geomorphology=”AL”)OR (Geomorphology =”OL”),”Landslide”,”No Landslide”)` ↵
- Click Show in the Raster Map Definition dialog box and OK in the Display Options dialog box.
- Inspect the results and close the map window.

12.1.2 Creating and running a script

The exact syntax for the script statements is something you know by heart only after working a considerable time with ILWIS. There are however several methods that you can use to create correct 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 Define. At that moment the expression for that specific operation is displayed on the Command line. You can then copy the expression from the Command line into the script. Within the script editor 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.

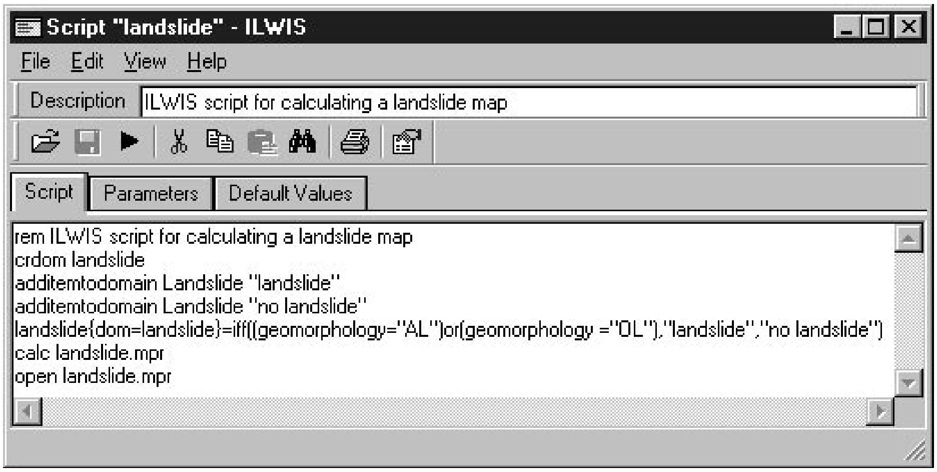
- The *ILWIS log file*. ILWIS keeps track of everything you are doing in a so-called *log file*. The ILWIS log file is called *Ilwis.log* and can be found in the Log file directory that is specified in the Directories part of the Preferences dialog box (File menu in the Main window). The *log file* is 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.

In the next exercise we will create a script that when executed creates the same landslide map as created from the Command line in the previous exercise.

☞

- Select Create Script from the File menu of the Main window. The Script editor (Figure 12.1) is opened.

The script that is needed to calculate the landslide map is shown below and as you can see that language is fairly similar to the Command line syntax. Instead of typing, you can also copy part of the statements from the Command line or from the *log file* into the script. Note that the filenames have changed so that we do not overwrite the objects of the previous exercise.



Description: ILWIS script for calculating a landslide map

```
rem ILWIS script for calculating a landslide map
crdom landslide
additemtodomain Landslide "landslide"
additemtodomain Landslide "no landslide"
landslide{dom=landslide}=iff((geomorphology="AL")or(geomorphology ="OL"),"landslide","no landslide")
calc landslide.mpr
open landslide.mpr
```

Figure 12.1: ILWIS Script editor with an example of an ILWIS script.

The line numbers in Figure 12.1 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 `Landslide`.
- In lines 2 and 3, two items are added to this domain.
- In line 4, the raster map `Landslide` 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 `Geomorphology`. The codes “`AL`” and “`OL`” stand for “`Active Landslide`” and “`Old Landslide`”.
- In line 5, the map `Landslide` is calculated. The `Calc` statement is a typical script language statement meaning that an object has to be calculated before proceeding to the next line of the script. This statement is used when the outcome of a calculation or a statement is needed as input for the next.
- In line 6, the map `Landslide` is displayed.

☞

- Type the following lines in the text box of the Script tab:

```
Rem ILWIS script for calculating a landslide map
Crdom Landslide
Additemtodomain Landslide “Landslide”
Additemtodomain Landslide “No Landslide”
Landslide{dom=Landslide} = IFF ((Geomorphology=“AL”)
OR(Geomorphology=”OL”),”Landslide”,”No Landslide”)
Calc Landslide.mpr
Open Landslide.mpr
```

- Click the Save button in the Toolbar of the Script editor and save the script as `Landslide`.

The script `Landslide` is created. Note that the first line starts with Rem. This indicates that this line contains a *remark* and will not be executed by ILWIS. To run the script:

☞

- Click the Run Script button ▶ in the Toolbar of the Script editor. When you already closed the Script editor, you can select the script in the Catalog, click it with the right mouse button and select Run… from the context-sensitive menu. You can also type `Run Landslide` on the Command line of the Main window.

The results of running this script are the map `Landslide` and the domain `Landslide`.

☞

- Check the contents of the raster map `Landslide` and the domain `Landslide` and close the map window, the domain and the Script editor afterwards.

12.2 Creating a script with calculations and expressions

When using MapCalc expressions in a script, 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.

The general syntax for MapCalc expressions in scripts is:
`Outmap = `*`Expression`*
Where,

`Outmap`	is the name of the output object.
`=`	is a definition symbol to indicate that a dependent output object is to be created, when the assignment symbol `:=` is used, an independent output object is created.
`Expression`	is an expression consisting of an operation name followed by the parameters required by this operation (between brackets, and separated by commas), or a MapCalc expression.

For example, to sum maps `Map1` and `Map2` to create `Map3`, type in the script:
`Map3 = Map1 + Map2`

When using TabCalc expressions in a script, it is required that the word `TabCalc` and the table name used for the expression are added to the syntax.

The general syntax for TabCalc expressions in scripts is:
`TabCalc `*`Tablename Expression`*
Where,

`Tabcalc`	is to indicate that the following is a table calculation syntax.
`Tablename`	is the name of the table used for the expression.
`Expression`	is the Command line syntax that you would use on the Command line of a table window to perform table calculations.

For example, to sum columns `Col1` and `Col2` in table `MyTable` and to store the results in column `Col3`, you can type in a script:

`TabCalc MyTable Col3 = Col1 + Col2`

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* (see Table 12.2) of the object after the object name:

`TabCalc `*`Objectname.ext`*` Col3 = Col1 + Col2`

! When you use long object names for maps or tables, and when the names of objects start with a digit, or start with or contain a space, or a special character, then these names must be enclosed in single quotes. The extension should be left outside the quotes. For more information, see the ILWIS Help topic, How to use long object names.

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

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 in a script. The script looks as follows:

	Rem ILWIS Script
1	`Geomorphology = MapRasterizePolygon(Geomorphology,Cochabamba.grf)`
2	`Landuse = MapRasterizePolygon(Landuse,Cochabamba.grf)`
3	`Landvalue = (Landuse.Landvalue) / 25`
4	`Landvalue1 = IFF(Slope > 20 , Landvalue * 0.7 , Landvalue)`
5	`Landvalue2 = IFF(Geomorphology = “OL”, Landvalue * 0.6, IFF ((Geomorphology = “AL”) OR (Geomorphology = “HE”), Landvalue * 0.4, Landvalue))`
6	`Landval_combined = MIN(Landvalue1, Landvalue2)`
7	`Landval_final = IFUNDEF(Geomorphology, Landvalue2, Landval_combined)`

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 `Geomorphology` and `Landuse` are rasterized, using the georeference `Cochabamba`.

- 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°, then the price of the land will only be 70% 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% of the average value. If the pixel is on an active landslide or on an active erosion area, the value is only 40% of the average. The information on landslides and erosion is stored in the map `Geomorphology`. Codes are used instead of the names of the geomorphologic units. The unit "`Old Landslide`" in the domain `Geomorphology` has the code "`OL`". If you use codes, the formulas can be much shorter.

- Now you have generated two maps that contain land values based on one criterion (`Landvalue1` and `Landvalue2`). What should you do for pixels where more than one of these criteria occurs? For example for pixels with a slope less than 20° which are located on an active landslide. The best approach 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 `Landval_combined` occupies a smaller area than the map `Landvalue` the formulas will result in undefined values, for those places where one of the input maps is undefined. This is corrected in line 7.

☞

- Double-click the script `Landvalue` in the `Catalog`. The `Script` editor is opened, in which you will see the script statements.
- Run the script by clicking the `Run Script` button ▶ in the `Toolbar` of the `Script` editor.

Since all the expressions in script `Landvalue` 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 (`Landval_final`) all previous maps are also calculated.

☞

- Open the map `Landval_final`. The calculation starts with the first map that was defined in the script. Have a look at the result and close the map window and the `Script` editor.

12.2.2 Example of a script for Table Calculation

In section 5.7 you have been practicing with table joining for an urban problem, using two tables: `Cityblock` (table linked to the map `Cityblock`, 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. Find 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 needed 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 table `Cityblock`. The information on the number of schools and the percentage of schoolchildren of the population is known per district and is stored in table `District`. Since you know for each city block in which district it is located, you can use the information from the table `Cityblock` and bring it into the table `District`. However, the table `Cityblock` 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 Cityblock.tbt`
2	`Tabcalc Cityblock Areadistrict = ColumnAggregateSum(Area, District, 1)`
3	`Tabcalc Cityblock Distrlanduse = District + Landuse`
4	`Tabcalc Cityblock Areadistrlu = ColumnAggregateSum(Area, Distrlanduse, 1)`
5	`Tabcalc Cityblock Residential {dom = perc;::1} = IFF(Landuse = “Residential”, 100 * Areadistrlu / Areadistrict, 0)`
6	`Tabcalc Cityblock Commercial {dom = perc;::1} = IFF(Landuse = “Commercial”, 100 * Areadistrlu / Areadistrict, 0)`
7	`Tabcalc Cityblock Institutional {dom = perc;::1} = IFF(Landuse = “Institutional”, 100 * Areadistrlu / Areadistrict, 0)`
8	`Closetbl Cityblock.tbt`
9	`Rem Open the table District`
10	`Opentbl District.tbt`
11	`Tabcalc District Residential = ColumnJoinMax(Cityblocks.tbt, Residential, District, 1)`
12	`Tabcalc District Rommercial = ColumnJoinMax(Cityblocks.tbt, Commercial, District, 1)`
13	`Tabcalc District Institutional = ColumnJoinMax(Cityblocks.tbt, Institutional, District, 1)`
14	`Tabcalc District Population = ColumnJoinSum(Cityblocks.tbt, Population, District, 1)`
15	`Tabcalc District Children = Population * Pchildren / 100`
16	`Tabcalc District Childpschool = Children / Schools`
17	`Closetbl District.tbt`
18	`Open District.tbt`

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

- In line 1, table `Cityblock` will be kept open in memory.
- 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 `Cityblock`.
- 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 the areas of the land use types 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 and that the precision for the values in the output column is set to 1.
- In line 8, the `Cityblock` table in memory is closed, and in line 10 the `District` table will be kept open in memory.
- In lines 11, 12 and 13, the percentage cover values for residential, commercial and institutional areas within each district as stored in table `Cityblock` are joined into table `District`. The aggregate function is needed, since 1 record of a district from table `District`, is linked to many records of the same `District` in table `Cityblock`.
- In line 14, another join operation is performed to join the population data from table `Cityblock`, summed up for each district, into table `District`.
- In line 15, the 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.
- In line 17, table `District` is closed.
- In line 18, the output table `District` is displayed.

☞

- Click the ILWIS button in the Toolbar of the Main window to make sure that all object types are selected.
- Double-click the script `Urban` in the Catalog. The Script editor is opened, in which you will see the script language.
- Run the script by clicking the Run Script button ▶ in the Toolbar of the Script editor.
- Have a look at the result and close the table `District` afterwards.

12.3 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. Any ILWIS object can be represented by parameters in a script as well as values and strings. Parameters in a script must be written as %1, %2, %3, …, %9 (see ILWIS Help, How to use parameters in scripts).

In this exercise you will calculate a slope map in percentages and degrees. The script looks like:

	Rem ILWIS Script: for creating a slope map
1	`Dem = MapInterpolContour(%1, %2)`
2	`Dem_dx = MapFilter(Dem, DFDX)`
3	`Dem_dy = MapFilter(Dem, DFDY)`
4	`%3.mpr = 100 * HYP(Dem_dx, Dem_dy) / Pixsize(Dem)`
5	`%4.mpr = RADDEG(ATAN(%3.mpr / 100))`
6	`Calc %4.mpr`

The line numbers in the table are not 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 contour vector map, indicated with parameter `%1`, is interpolated using georeference `%2` to create a Digital Elevation Model `Dem`.

- In line 2, the digital elevation model is filtered using a DFDX filter, creating a new map `Dem_dx` containing the first derivative in x-direction (df/dx) per pixel.

- In line 3, the digital elevation model is filtered using a DFDY filter, creating a new map `Dem_dy` containing the first derivative in y-direction (df/dy) per pixel.

- In line 4, a slope map in percentages `%3` is calculated.

- In line 5, a slope map in degrees `%4` is calculated.

- In line 6, output map `%4` is calculated and therefore all maps created by the script are calculated.

Note that this script contains four parameters: %1 till %4.

When you run a script with parameters these parameters should be defined either before or when the script is run. The parameters can be defined in the following ways:
- In the Script editor you can fill out the Parameters tab. Parameter Name and Type can be entered here and will later on appear in a Run Script dialog box when a user runs the script. The user can then select object names for input parameters and/or type object names for output parameters.

- When the Parameters tab is used, also default values for the input and output parameters can be filled out in the Default Values tab. This tab is very useful when you want to test your script or when you directly want to run your script.

- When no parameters are defined on the Parameters tab, you can run the script from the Command line and specify the input and output objects that should be used for the parameters after the script name:

  `Run` *`Scriptname Parameter1 Parameter2 … Parameter9`*

  ILWIS will replace every parameter definition (%1….%9) by the specified parameter.

For this exercise the parameters of Table 12.3 are used.

Table 12.3: Script parameters.

Parameters for the Slope script			
Parameter	**Name for Run Script dialog box**	**Type**	**Optional default object**
%1	Input Contour map	Segment Map	`Contour.mps`
%2	Georeference to be used	Georeference	`Cochabamba.grf`
%3	Output Slope map in Percentages	Filename	`Slopepct.mpr`
%4	Output Slope map in Degrees	Filename	`Slopedgr.mpr`

When run, the script will be interpreted as follows:

```
Rem ILWIS Script: for creating a slope map
Dem = MapInterpolContour(Contour.mps, Cochabamba.grf)
Dem_dx = MapFilter(Dem, DFDX)
Dem_dy = MapFilter(Dem, DFDY)
Slopepct.mpr = 100 * HYP(Dem_dx, Dem_dy) / Pixsize(Dem)
Slopedgr.mpr = RADDEG(ATAN(Slopepct.mpr / 100))
Calc Slopedgr.mpr
```

We will first run the script from the Command line of the Main window.

- Open the script `Slope` and look at its contents.
- Close the Script editor.
- You can run the script by typing the following expression on the Command line of the Main window:
  `Run Slope Contour Cochabamba Slopepct Slopedgr` ↵
- Have a look at the resulting maps `Slopepct` and `Slopedgr`. Note that the calculations start when you open the maps.

We will now enter the parameters in the Parameters tab and run the script from the Script editor.

☞

- Open script Slope and click the Parameters tab. Set the number of parameters to 4.
- Enter the parameter Names and Types according to Table 12.3.
- Enter Run Slope Script on the Description line.
- Save the script.
- Run the script by clicking the Run Script button ▶ in the Toolbar of the Script editor.
- The Run Slope Script dialog box now opens. Select the correct input map and georeference and type for the Output Slope map in Percentage Slopepct2 and for the Output Slope map in Degrees Slopedgr2.
- Click OK.
- Have a look at the resulting maps Slopepct2 and Slopedgr2.
- Click the Default Values tab. Select the Input Contour map and the Georeference to be used according to Table 12.3. Type for the Output Slope map in Percentage Slopepct3 and for the Output Slope map in Degrees Slopedgr3.
- Save the script.
- Run the script by clicking the Run Script button ▶ in the Toolbar of the Script editor.
- Have a look at the resulting maps Slopepct3 and Slopedgr3. Close the map windows and the Script editor.

12.4 Running a script from another script

In this example, we will make a new script DensIn which will run the existing script Density three times, each time using a different map as input.

Script Density calculates the density of landslides within certain units. It reads:

	Rem Script Density to calculate landslide density
1	`s%1 = TableCross(%1,Slide)`
2	`calc s%1.tbt`
3	`tabcalc %1 AreaClass = ColumnJoinSum(s%1.tbt,Area,%1)`
4	`tabcalc s%1 AreaSl = iff(Slide="landslide",Area,0)`
5	`tabcalc %1 AreaSlide = ColumnJoinSum (s%1.tbt,AreaSl,%1)`
6	`tabcalc %1 Density {dom=perc} = 100 * AreaSlide/AreaClass`
7	`calc %1.tbt`

As before, the line numbers are only used to explain the script. In short:
- A cross table is calculated from a variable input map (%1) and raster map Slide.
- In the cross table, the total area of each class is calculated and written into the attribute table of the variable input map (AreaClass).
- Then, if landslides occur, the total area with landslides is calculated per class, and written into the attribute table of the variable input map (AreaSlide).
- In the attribute table, the density of landslides is calculated by dividing AreaSlide by AreaClass.

With script DensIn we can now call script Density and use various input maps:

	Rem Script DensIn that serves as input for the Density script
1	`Run Density Geology`
2	`Run Density Slope_classes`
3	`Run Density Catchment`

- In line 1, script `Density` is executed using map `Geology` as parameter `%1`.
- In line 2, script `Density` is executed using map `Slope_classes`.
- In line 3, script `Density` is executed using map `Catchment`.

☞

- Create a script `DensIn` and enter the lines as given in the example.
- Save the script and exit the editor.
- In the `Catalog`, click script `DensIn` with the right mouse button and choose `Run`.
- Have a look at the resulting attribute tables `Geology`, `Slope_classes` and `Catchment`.
- Close the tables after you have seen the result.

12.5 Special script language

Even though the script language is not intended to be a programming language, but merely a tool to help you process data, it does have special features that makes it easier to work with and to give it some extra capabilities. Some examples of the special script language is given in Table 12.4, a complete overview is given in the ILWIS Help topic, Appendices ILWIS script language (syntax).

Table 12.4: Some commands that can be used on the Command line.

Command	Example	Description
Begincomment Endcomment	begincomment Open `Geomorphology.mpr` endcomment	All lines of text between the commands begincomment and endcom ment are ignored by the script.
Pause *seconds*	Pause 20	Stop the script for a certain amount of time (seconds).
Message *text*	Message `Click to continue`	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.
Open -noask *object.ext*	Open -noask `Cityblock.mpa`	Opens the object *object.ext* with its default display options
Closeall	Closeall	Close all ILWIS windows except the ILWIS Main window.

Summary: Scripts

- Scripts are used to automate the operations in ILWIS.

- A script is a list of commands, calculations 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 as listed in the Appendices, ILWIS commands and expressions section of the ILWIS Help: opening dialog boxes, MapCalc, TabCalc, performing operations and some other actions.

- A script consists of an object definition file with extension `.isl` (ILWIS Script Language) and a data file with extension `.isf` (ILWIS Script File).

- A script can be made by copying an expression from the Command line after you have filled in all required parameters in the dialog box of a certain operation, and clicked Define. At that moment the expression for that operation is shown on the Command line. You can copy this expression from the Command line into your script.

- A script can also be made by copying parts from the *ILWIS log* file after you have executed some operations 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 you can find its directory from the Preferences. The *.log* file can be opened with any text editor.

- A script can be started from the Script editor, by choosing the Run … command on the script's context-sensitive menu in the Catalog or from the Command line of the Main window by typing: `Run Scriptname` ↵

- Any ILWIS object can be represented by parameters in a script as well as values and strings. Parameters in a script must be written as `%1`, `%2`, `%3`, etc. Parameters have a Name and Type that are entered on the Parameters tab so that a user can select objects for the parameters in a dialog box when running the script.

- You can also specify default objects for parameters on the Default Values tab.

- Other scripts and other Windows applications can be called from within a script.

- A range of special script language is available to give a script more possibilities.

12.6 Functions

In Chapter 5 you have seen the use of calculation formulas to work with tables, and in chapters 7 and 8 those to work 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 ILWIS 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 will 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 expression on the Command line of the Main window: `Result1 = IFF(Dem > 4000, 10, 0)` ↵
- Click Show, evaluate the result and close the map window.

It means: If a pixel in map `Dem` (Digital Elevation Model) has a value greater than `4000`, 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 expression on the Command line:
  `Map1 = IFF (Dem = 1, 1, 1) ↵`
- The Raster Map Definition dialog box is opened. Select system Domain `Value`, and set the Value Range from `0` to `1`, and the Precision to `1.0`. Click Show.
- Type the following expression on the Command line of the Main window:
  `Random = 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. `Random` 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 system Domain `Value`, and set the Value Range from `0` to `1`, and the Precision to `1.0`. Click Show.
- Examine the results, and then 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 expression on the Command line:
  `Min3 = MIN(Tmb2, Tmb3)↵`
- The Raster Map Definition dialog box is opened. Select system Domain `Image` and click Show.
- Examine the results, and then close the map window.

User-defined functions

fn

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 and the pocket line calculator. Especially when you need to execute calculations that require a lot of typing work several times, user-defined functions may be time saving. A user-defined function is an expression that may contain any combination of operators, functions, maps and columns in tables.

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 for the Function Name: `Average_2`.
- Type for the Expression: `(a + b)/2`.
- Type for the Description: `Averaging two value maps`. Click OK.

The Function editor appears showing your newly created function. If necessary, you can edit your function. 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 Average_2(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 are listed 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.

-In line 3, the actual function is given. Note that the expression ends with a semicolon (;).

-In line 4, the end of the function is indicated with the word `End`.

Now you can use your function on the Command line of the Main window or table window. Type an expression that starts with an output map name (or column name) followed by the definition symbol (=), then the name of your function and fill out the parameters. The parameters, replacing the characters *a* and *b* in your function, have to be entered in brackets separated by commas. The parameter that is filled in first is taken as the first parameter encountered in your user-defined function.

☞

- Close the Function editor.
- Type the following expression on the Command line:
  `Average_tmb=Average_2(Tmb2, Tmb3)↵`
- The Raster Map Definition dialog box is opened. Select the system Domain Image and click Show.
- Examine the results in map `Average_tmb`.
- Close the map window.

You can use this function also on the Command line of a table window to calculate the average of two value columns.

The following example shows a more complex expression. We will calculate the direction of slopes and create an 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: `Dx` and `Dy`. Later, when we apply the function, you substitute the `Dx` and `Dy` parameters 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 Function editor is opened.

As you can see in the first line of the function definition, 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 declaration `Value 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 so that it looks as above.
- Save the function and close the Function editor.

To apply the function Aspect:

☞

- Type the following expression on the Command line of the Main Window:
  `Aspect = Aspect(Dem_dx, Dem_dy)` ↵
- The Raster Map Definition dialog box is opened. Select system Domain Value, and set the Value Range from 0 to 360 and the Precision to 1. Click Show.
- Examine the results in map Aspect and close the map window.

Now you can easily calculate several aspect maps of other areas. You only have to fill out your new input variables for the function; you have to specify the new Dx and Dy map names of the other area.

Summary: Functions

- ILWIS contains over 50 different functions that are pre-programmed and that can be used in Table Calculation, in Map Calculation, Script and 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 columns.

- User-defined functions are especially useful when you need to execute certain calculations, which require a lot of typing effort on the Command line.

- To apply your function, type an expression 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 fill out the function parameters. The parameters, replacing the characters *a, b, c,* etc. in your function, have to be entered in brackets separated by commas. The parameter filled out first is taken as the first parameter encountered in your user-defined function.