



Red7 Process



**Processing, analysis and indicators
generation**

CLASS MATERIAL 2

REDATAM© is a software developed by CELADE (Latin American and Caribbean Demographic Center), Population Division of the Economic Commission for Latin America and the Caribbean (ECLAC) of the United Nations
www.cepal.org/en/topics/redatam

Table of Contents

Table of Contents.....	1
Starting a session in Red7 Process Module	2
I. Redatam7 Fast & Friendly	2
II. The Database Dictionary	4
III. The Project (work space).....	4
IV. Programming: generating outputs	5
V. Using filters in the output table: the clause FOR.....	9
VI. Using filters: the clause UNIVERSE	10
VII. Geographic Area Selection	11
VIII. Deriving new variables – The RECODE Command	14
IX. Promoting information - The COUNT command.....	16
X. Promoting information - The SUM function and The OPTIONS WEIGHT function	17
XI. Using the DEFINE/SAVE command	18
XII. The SWITCH function	19
XIII. Using logical expressions in arithmetic expression	21
XIV. Creating new indicators	23
Defining a map composition.....	34
XV. Procedure.....	34

Starting a session in Red7 Process Module

I. Redatam7 Fast & Friendly

REDATAM is the result of the efforts undertaken by CELADE - Population Division of ECLAC in order to allow data recovery for small areas by microcomputer through a friendly and interactive software solution that enables access to hierarchical structured large data files including microdata from population and housing censuses, agricultural censuses, household statistics, and indicator systems and surveys in general.

REDATAM organizes and stores all the information in these files so that the user can quickly obtain any tabulation or other statistics to the smallest defined in the data hierarchical area such as an apple segment establishment or grouping of these hierarchical units.

In the early 1980s, the penetration and use of personal microcomputers in Latin American countries began to grow. Early on, they were much slower than mini and mainframe computers, but with the CELADE-developed strategy of reading the file variables in reverse already receiving recognition, it was possible to accelerate processing times, with programs obtaining a specific tabulation by only accessing required and chosen variables and entries. This processing logic, in addition to the advances made in personal computer technology, opened up a new horizon in the development of census microdata processing tools.

With this step in computer development, on top of the potential for censuses to produce information geographically disaggregated into small units -- of people, of households, and of dwellings in a given country -- the REDATAM software developing project was born. The name comes from the acronym for **RE**trieval of **DA**ta for small **Ar**ea by **MI**crocomputers. The philosophy behind the development of this software was to offer a user-friendly tool that was accessible to all public policy and population researchers and decision makers.

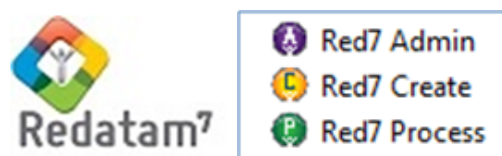
Originally programmed for the DOS operating system, REDATAM used a simple programming language of indicators and tabulation generation, expanding access to a broader group of users, in addition to computer programmers.

To optimize use of census or other data, REDATAM databases are normally built with microdata, that is, with variables which refer to individuals, households or other analytical elements; these variables can then be combined in tabulations to produce findings for any geographic area, as defined by the user. The data are organized in a hierarchy, making it possible to derive, for example, new household variables based on the number or characteristics of the individuals who live in each home. It is possible to select any series of subareas to process only the chosen data subset, accelerating, in turn, the aggregate calculation needed to generate various types of tabulations. Furthermore, any derived variable can be stored permanently in the database for future use. No prior programming experience is needed to quickly obtain tabulations and other statistical results, since it has a graphical user interface. The program contains resources for processing and creating maps connected to database geographical levels. This makes it possible for users to see on-screen a spatial analysis of the aggregated statistics that the system produces in any geographical area. The visualization functions, in addition to data manipulation and high-speed processing, provide fast access to data and increase their value. Its Web Server modules, moreover, make it possible for users to access online microdata processing, thus facilitating information dissemination.

Versions of REDATAM

Redatam version	Generation	Publication	OS	Census round
Redatam DOS	First generation	1985	DOS	1980
Redatam 3.1	First generation	1986	DOS	
Redatam-Plus	Second generation	1991	DOS	1990
winR+	Third generation	1997	Windows	
Redatam+G4	Fourth generation	2001	Windows, WebServer	2000
Redatam+SP	Fourth generation	2004	Windows, WebServer	
Redatam7	Fifth generation	2014	Windows, WebServer, 64 bits, Unicode-multilanguage	2010

Redatam7: speed, efficiency and ease of use



During these 25 years of the REDATAM Project, enormous jumps in computer technology have taken place (DOS operating system, Windows -- 16 bits, then 32 bits and currently 64 bits --), which CELADE, the Population Division of ECLAC, with the support and funding of various donors, has addressed by providing countries with a suitable and up-to-date tool. For this reason, we are presenting the new generation of REDATAM, called REDATAM 7, which incorporates new development technology -- based on C++, Delphi, Java and JavaScript -- and which, with the original program's logic and organizational structure intact, strives to improve processing speed and indicator programming, as well as make the user's interaction with the different REDATAM modules easier. Currently it is only available for the Windows platform.

REDATAM 7 improvements

A noteworthy development in these improvements has been the significant contributions of the network of users in Latin American countries and in other parts of the world, who have conveyed to us their expectations for and difficulties with using previous REDATAM generations; with that in mind, our primary improvements are:

1. **The standardized use of the XML language:** making it possible to synchronize documentation tasks, training and programming, as well as the interconnection with other, country-required computer tools, while shrinking the gap between development and user know-how.
2. **A newly designed and programmed compiler:** starting with the definition of a grammar or syntax that adjusts the language to new requirements, improving detection errors and deployment of error messages in REDATAM language.
3. **Unicode Support:** for generating dissemination applications in other languages used in the region -- Quechua, Creole, Guarani -- and of other Arab and Chinese regions, etc., in addition to the already existing languages: English, Spanish, Portuguese, French and Bahasa (Indonesia).

4. **Presentation of user defined tabulations:** access to each one of the elements involved in tabulation presentation, defining which ones to present and their position; the outputs are presented in XML for export to other applications.
5. **Unlimited number of dimensions:** currently the maximum number of variables that can be included in a tabulation are five plus an area break; with the newly designed REDATAM engine, tabulations may have unlimited dimensions.

II. The Database Dictionary

The database dictionary (.dicx file) is the main object from where the user can manipulate, analyze and browse the data contained in a Redatam database. The first thing to do in a session is to open a dictionary file from the main menu by clicking **File>Open Dictionary**. The Red7Process module is installed with a census database for a fictitious country called Nueva Miranda and all the examples and exercises in this manual are based on the Nueva Miranda dataset. To open the database dictionary for Nueva Miranda, open the file C:\Users\Public\Documents\redatam7 demo \NMIR\BaseR\NmirEng.dicx

The left side of the dictionary window displays the hierarchical database structure as a directory-like tree of *entities*, each of which has individual elements or members, which are often geographical, like provinces, districts, or city blocks, but can also be individual persons, houses, schools, etc. The right-side box contains the list of *variables* describing the currently highlighted entity on the left side. Click on another entity name to change the current entity. The list of variables in the right box is then changed accordingly. The dictionary can be laid out with the variables appearing to the right of the entity tree (by selecting **Layout Presentation>Vertical**) or below the entity tree (by selecting **Dictionary>Layout Presentation>Horizontal**).

Double click on a variable name (in the list on the right) to open a window displaying information about the variable. Click on the plus or minus sign to the left of the entity name (or label) in the entity tree in order to either expand or collapse the branches beneath this entity.

III. The Project (work space)

In Red7 Process the concept of handling documents and folders within a “project” is introduced. In the project window the user can define the database, the working directory, the table styles, several groups of program files, groups of selections, or other related documents.

Browse each tab to define working folders or grouping names to handle Redatam programs.

IV. Programming: generating outputs

THE BASIC COMMAND SET

Redatam has two levels of programming to get results from a database: a) Fast Results by using the easy tables; b) Free Editor. The first level is designed for you to get results using only the mouse, in a "point and click" fashion, requiring no programming skills at all. However, this level has some limitations because it does not provide the functionality to define new derived variables. The second level is based on a structured syntax that the user can write in the editor window in order to get new variables, indicators or tables. The results can be obtained through programming a command set. In a command set, the user tells Red7 to carry out processes to create new variables and produce specific tabulations and other results via a set of commands from the Redatam language.

The programs are stored in files with the extension .spc (Statistical Processor Commands). To create a new program, or open an existing program, choose the function File in the main menu.

Then you can select New for a new object, or Existing for an existing object. In each case (New or Existing), choose Command Set. In the case of a new program, the system opens directly the window for free editing. In the case of an existing program, the system opens a dialog box (within the project window) to navigate and select the desired program. To create a new program, or open a program, it is necessary to have an active database, in other words, the user must already have opened a database dictionary. Otherwise, the File menu will not permit the user to select either New or Existing.

There are two ways of programming in Red7:

- a) Using the Programming Assistant; or
- b) Writing the program directly in the Command Editor which provides hints and shortcuts as well.

Programs are often written using a mixture of these techniques. Programs can be written directly in the Command Editor. In addition, the Programming Assistants can be accessed by right clicking within this editor and they can be used to aid in the writing of the commands. A Red7program can be created in one way (for example using Assistants) and later worked on in another way (by writing directly in the Command Editor).

The language has three basic commands:

RUNDEF to define the environment in which a process will occur, including the specific selection set to be used or/and the universe to filter cases. A Program must start with a RUNDEF, specifying the Selection and the UNIVERSE filters, if needed. If none is specified, the RUNDEF section can be omitted. It is always the very first command and it must be unique in the program. This section RUNDEF can be edited through the RUNDEF Assistant.

DEFINE to create new variables and indicate their nature and scope.

TABLE to indicate what the specific process and output table should be generated.

Each of these main instructions can be qualified by subordinate clauses. For example, to DEFINE a new variable, the user can specify its nature via clauses including AS, TYPE, RANGE, VARLABEL, VALUELABEL, limit the cases to which it applies by a FOR clause (which is similar to an “if” in Redatam-Plus), and RECODE the values, QUANTIFY and COUNT elements, etc.

```
DEFINE housin.newvar
AS COUNT person
TYPE INTEGER
VARLABEL “Total persons in the house”
```

Similarly, the output can be specified with the TABLE command using AS <tabletype>, where tabletype may be FREQUENCY, AVERAGE, CROSSTABS, AREALIST, etc., OF a set of variables, and further specified by clauses beginning with FOR, etc.

```
TABLE tab1 AS FREQUENCY OF housin.newvar
```

In Redatam7 the syntax for each section can be reduced to a minimum, for example, the above table section can be written as:

```
FREQ newvar
```

If the variable used in the table “newvar” is unique in the database, there is no need to add the entity were it belong.

Scheme

A simple Program using RUNDEF and TABLE

Write a program to get the same results as the frequencies, crosstabs and averages that are produced by the Wizards.

Procedure

- 1 Opening the Command Editor: To open the Command Editor screen (with a database already active), go to the main menu, select **File**, and then choose **New**.
- 2 Save: It is recommended to do a Save from time to time, before executing the program and when you finish your job. Click on the save icon in the top left-hand corner of the Command Editor and give the program a meaningful name, such as Prog1Freq. You can use also the save icon in the Red7toolbar, or use CTRL + S.
- 3 RUNDEF, always the first command: Accessing the RUNDEF Assist: After opening the Command Editor, you can use the specific Assistants by means of the popup menu, with the right mouse button. Select the whole RUNDEF command by positioning the mouse before the R of RUNDEF, and holding the mouse left button, move over the two lines of the command until the end of the word ALL, and then press the right mouse button.
- 4 Fill in the spaces in the RUNDEF Assistant:

Name, accepts any text.

Selection, defines the area that you want to process. By default it will read all the database (ALL). If you want to use a specific selection, you should have already created the selection file (with extension .slw). Use the browse button to find and select it.

Universe, optional. This clause defines the filter expression to select the cases to process.

Go back to the Command Editor with a click in the **OK button** in the bottom part of the Assistant screen, or use the Save button in the Toolbar.

- 5 Set the table command position in the program (TABLE): Click the mouse in the next free line after the RUNDEF command. If there is no free line, click the mouse at the end of the second line of the RUNDEF command and press the [ENTER] key, and then click the mouse in the new line. This will be the place where you will put the first TABLE command. Once you finish with the TABLE Assistant (see next step), the command will be inserted into the Window in the same place where the cursor is—if you want, you can leave an empty line after the RUNDEF command to improve its readability.
- 6 Use the TABLE Assistant: with the Command Editor as the active window, that is, its title bar being a darker shade of blue, call the popup menu (right mouse button) and select the TABLE Assistant. In the first tab of the TABLE Assistant, the Table tab, the user chooses a name for the table and the kind of table they want, for example a frequency table (TABLE AS FREQUENCY). Select the Frequency tab and then select variables by clicking and dragging from the variable list contained in the dictionary window of your active database.

Demonstration

Compute the distribution of population by age and sex and the distribution of dwellings by type in the Nueva Miranda database: Open a new Commands Editor and type the following program:

```
TABLE Freq1
AS FREQUENCY
OF PERSON.EDQUINQ, PERSON.SEX, HOUSIN.TYPHOU
```

Run the program and check the results

Notes:

1. The RUNDEF section is omitted because there is no special selection file. The RUNDEF/SELECTION clause allows you to select a different geographical area to process. By default it uses ALL database.
2. "Freq1" is the name of the table. It could be any user defined name.
3. If you write the complete syntax for the TABLE it will require a name.
4. The clause AS is followed by the type of table required.
5. The clause OF is followed by the list of variable identifiers to tabulate.
6. A variable identifier is always built from the entity name (PERSON) followed by a dot immediately followed by the variable name (EDQUINQ). NO BLANK (space) must appear in the variable (PERSON.EDQUINQ).
7. Optionally use the Assists windows to build the table command.
8. Type of tables are: FREQUENCY, CROSSTABS, AVERAGE, MEDIAN, AREALIST, MATRIXOP, VIEW

Exercises

1. Create a frequency table of the variables OWNERS, ROOF, TROOMS, and TOILCO.
2. Compute the sex distribution by Kinship Relationship
3. Compute the sex distribution by Kinship Relationship and Marital Status.
4. Compute the average age by sex and Marital Status. Comment on the results.

V. Using filters in the output table: the clause FOR

To restrict the scope of the tabulation command or the definition of a variable we use the clause FOR followed by a logical expression.

The result of a logical expression is of BOOLEAN type: TRUE or FALSE. The logical expression is evaluated and only those for which the logical expression is TRUE are included in the TABLE.

Building a logical expression requires the use of the logical operators:

= equal
> greater than
< less than
<> not equal than
>= greater or equal than
<= less or equal than

And the relational operators: NOT AND OR

Syntax

```
TABLE table1
  AS FREQUENCY
  OF <variable list>
  FOR <logical expression>
TABLE table2
  AS CROSSTABS
  OF <var list> BY <var list> [ BY <var list> [ BY <var list> ] ]
  FOR <logical expression>
TABLE table3
  AS AVERAGE
  OF <variable > BY <var1> [ BY <var2> ]
  FOR <logical expression>
```

Demonstration

Computing the average household head age by sex

```
RUNDEF EasyCross
  SELECTION ALL

TABLE Table1
  AS AVERAGE
  OF PERSON.AGE BY PERSON.SEX
  FOR PERSON.RELAT = 1
```

Exercises

1. Compute the average age of household heads by Sex and by Type of Family 1.
2. Compute a Crosstabs between type of fuel used for cooking and type of housing for home owners only i.e. excluding renters and other types of ownership.
3. Compute a Frequency of Type of Activity (ACTYPE) for female household heads and in the same command set compute a Crosstabs between Type of Education and Marital Status for single female household heads.

VI. Using filters: the clause UNIVERSE

The UNIVERSE clause, which can be used only in the RUNDEF command, restricts the entire run to cases that comply with the specified Boolean expression, i.e. when the expression is true. It is use as a global filtering. The restriction applies to all the outputs of a Run.

Syntax

```
RUNDEF <runid>  
    SELECTION <selection>  
    UNIVERSE <logical expression>
```

Demonstration

Compute the sex distribution only of the population that described their type of activity as 'Housewife'.

```
RUNDEF program3  
    SELECTION ALL  
    UNIVERSE PERSON.ACTYPE = 6  
  
TABLE one  
    AS FREQUENCY  
    OF PERSON.SEX
```

Notes

1. The logical expression must be defined only with variables that exist permanently on disk; it cannot include DEFINED variables from the current command set.
2. The Keyword **FOR** or **FILTER** can be used instead of **UNIVERSE**.
3. If a DEFINED variable will be saved in the command set the UNIVERSE cannot be applied.

Exercises

1. Compute the average age of women who had children in the last year (CHILDY).
2. Compute the proportion of children aged 11-16 who attend school.

VII. Geographic Area Selection

Since the user seldom wants to process the entire data set with REDATAM+SP, the user can identify a selection set that defines the specific elements (usually geographic areas) that will be processed to obtain statistical results. Any number of different selection sets may be defined for a given Red7 database and each selection set can be built from any combination of areas to define the final selection.

The area selection provides Red7 with a means to target specific areas of the database. Traditionally, the selection of a sub database was related to the geography, for this reason the sub database is called an area selection.

Steps to create a selection

- ❖ Open the selection window by choosing **File>New** and click on the down arrow to choose **Selection** from the menu.
- ❖ The system displays the selection screen with the database structure, showing the root entity, that is, the most aggregated level in the database.
- ❖ Use the expand option, either from the popup menu or from the toolbar that is also shown. The hierarchical structure "opens" at its second level.

The toolbar buttons are the same as the options that appear in the popup menu.

The elements in this second level (generally they correspond to geographical entities) are displayed with their identification codes, their element names, or both.

- ❖ Click in any element and repeat the process to expand it into its lower elements.
- ❖ Locate the cursor over the element you want to select.
- ❖ Mark the element with a double click, or using the select option in the popup menu, or the toolbar to select the element. The small square at the left side of the element should become yellow.
- ❖ Repeat the selection process for all those elements you want to include in the new selection set.
- ❖ If you select an element that was not expanded into its lower elements (does not have the symbol on its side), all the lower elements under this level will be selected.
- ❖ To collapse all the lower elements and branches, click on the minus sign to the left of an element and all lower elements and branches will be contracted.
- ❖ To save the selection set, use the save icon in the selection window toolbar or the popup menu. Use the save dialog box to find the directory where you want to save the file and give it a name (.slw extension by default).
- ❖ A double click on a selected element deselects it.

QUIZ 1

Name: _____

Date: _____

Please answer the following questions

1. What proportion of females between the ages of 25 and 35 years old have had children?

2. Calculate the average age of female heads of household.

3. What is the distribution of the population by Type of Education?

4. What is the range defined for the variable Type of Housing?

5. How many branches does this database contain?

6. How many households don't have piped water in their homes?

7. Find out the number of male headed households of 45 years old.

QUIZ 2

Name: _____

Date: _____

Please answer the following questions

1. Compute a cross tabulation of Type of Education by age group for female household heads over 15.
2. Compute a cross tabulation of School Attendance by age for women between 6 and 18.
3. In which counties do you find the greatest number of school pupils?

4. What is the difference between the UNIVERSE and FOR clauses?

5. Write a command set that computes the distribution of mothers by age group?

8. What is the distribution of Type of Activity for single female-headed households?

VIII. Deriving new variables – The RECODE Command

The DEFINE command is used to derive new information from the database. This is stored in a temporary variable and its definition involves various types of operations such as summing, recoding, arithmetic or logical expression, reading data from external source etc.

The RECODE command is used to create a new variable based on the recoding of an existing one.

Syntax

```
DEFINE entity.newvar
      AS RECODE <variable> < recoding scheme>
      TYPE INTEGER
      RANGE min - max
```

<u>Recode scheme format</u>	<u>Example</u>
• (<previous value> = <new value>)	(3 = 1)
• (<previous value1> TO <previous value2> = <new value>) or equivalently (< previous value1> – <previous value2> = <new value>)	(2 TO 5 = 3) (2 – 5 = 3)
• (LOWEST TO <previous value> = <new value>)	(LOWEST TO 4 = 2)
• (<previous value> TO HIGHEST = <new value>)	(11 TO HIGHEST = 5)
• ELSE <newvalue>	(12-20 = 12) ELSE 13

Notes

1. Values not included in the recoding scheme keep their original value
2. The ELSE keyword re-assigns all ranges not previously assigned
3. Variable type is always INTEGER
4. The RANGE value must fit with the maximum & minimum of the recoding scheme

Demonstration

One reason for carrying out this operation might be the need to create a new variable grouping people according to their age. The first group could be the children, the second group the adults and the third group the elders, for example those from 0 to 14 as children; 15 to 64 as adults; and 64 and over as older persons.

A new variable is created taking the value 1 for persons from 0 to 14, 2 for persons 15 to 64 and 3 for persons 64 and over.

The Red7 syntax to create and tabulate this variable would be as follows:

```
RUNDEF job  
SELECTION ALL
```

```
DEFINE PERSON.AGEREC3  
AS RECODE PERSON.AGE  
(0 - 14=1)(15 - 64=2)(64 - Highest = 3)  
TYPE INTEGER  
RANGE 1 – 3
```

```
TABLE Tab1  
AS FREQUENCY  
OF PERSON.AGEREC3
```

NOTE:

1. The variable type must be included in the definition. In this particular case, the type of this new variable is obviously an integer variable.
2. The range must be also included in the definition. Since, the variable can take only 3 values; the range is 1 to 3.

Exercise

1. Recode age into three broad age groups.
2. Recode households into those which use either gas or electricity for cooking and those that use other fuels.

IX. Promoting information - The COUNT command

The DEFINE...COUNT command returns the actual number of elements belonging to an entity. For example the number of persons in a household is computed:

```
household n      (owner entity)
  person 1      (counter entity)
  person 2
  person 3
  person 4
```

The resulting variable contains 4.

Syntax

```
DEFINE owner.totcount
  AS COUNT lowerent
  TYPE INTEGER
```

where **owner** is the owner entity, **totcount** the resulting variable and **lowerent** the counted entity.

Notes

1. The “counted” entity may be of any level below the owner entity. It is therefore possible to count the total number of persons by household or the total number of persons within any geographic area.
2. The “counted” entity must belong to the same branch of the database.
3. The TYPE of the resulting variable is always an INTEGER variable.
4. A “counted” variable is always a post-process variable

Exercises

1. Find out the largest population by District in Nueva Miranda
2. Does this District also have the largest number of households?

X. Promoting information - The SUM function and The OPTIONS WEIGHT function

Is used to sum the values of a variable at a lower entity level to a new variable defined at a higher entity level, taking into account only elements in the Selection set and only those that pass the DEFINE...FOR or RUNDEF...UNIVERSE filters, if any.

Demonstration

```
DEFINE HOUSIN.SUMAGE
  AS SUM PERSON.AGE
  TYPE INTEGER
```

The same result can be obtain using the OPTIONS WEIGHT function

```
DEFINE HOUSIN.SUMAGE
  AS COUNT PERSON
  OPTIONS WEIGHT PERSON.AGE
```

So the command set to produce these results would be:

```
RUNDEF job
  SELECTION ALL
```

```
DEFINE HOUSIN.HAGE
  AS SUM PERSON.AGE
  TYPE INTEGER
```

```
TABLE TT
  AS FREQUENCY OF HOUSIN.HAGE
```

```
DEFINE HOUSIN.HAGE2
  AS COUNT PERSON
  OPTIONS WEIGHT PERSON.AGE
  TYPE INTEGER
```

```
TABLE TTT
  AS FREQUENCY OF HOUSIN.HAGE2
```

Notes

1. If there are three persons in a house, aged 25, 35, and 42, the value of HOUSIN.SUMAGE for this house would be 102.
2. The variable to be summed must normally be quantitative such as age, number of rooms in the dwelling etc. The same cautions involving hierarchical processing apply to SUM.
3. Do not use parenthesis with the SUM clause, since it gives a compile error.

Exercise

1. Calculate number of households with no water piped for each COUNTY (give the results in number of persons).

XI. Using the DEFINE/SAVE command

When a derived variable is of general interest or when it is used repeatedly, one should consider saving the variable permanently in the database. It then becomes part of the dictionary and can be used as an original variable.

Syntax

```
RUNDEF saving
      SELECTION ALL

DEFINE HOUSIN.HHAGE
      AS PERSON.AGE
      FOR PERSON.RELAT = 1
      TYPE INTEGER
      RANGE 0-99
      VARLABEL "Household Head Age"
      SAVE HHAGE  OPTIONS OVERWRITE
```

Notes

1. In order to be saved a variable must be computed for the entire database. Thus the only SELECTION permitted is ALL (see RUNDEF)
2. The default directory where the bin file is stored is the current working directory.
3. When the variable must be replaced, it must be first deleted from dictionary. Use the OPTION OVERWRITE always.

Demonstration

Create and save as a new household variable the total number of persons in the household; and as a new district level variable the total number of households in the district

Exercise

1. Compute and save the household level variable: sex of the head of the household.
2. Compute and save a household variable showing the total number of persons under 14 years old in the household.

XII. The SWITCH function

The SWITCH clause is very useful to define a new variable depending upon a series of conditions and expressions that requires the evaluation of more than one variable, using the form if ... then ...

As far as the definition of a new variable is concerned, this is the most powerful clause in Red7 , because it can use multiple variables and logical expressions to obtain the final result to be assigned to one solely new variable.

Examples

- a) You have to create a new variable that will classify households according to the age and sex of the household head. The new variable will be called "Sex_Age" and its value will be determined based on the following person level variables: Kinship Relationship, Sex and Age.

```
DEFINE HOUSIN.AGE_SEX
AS SWITCH
INCASE PERSON.RELAT = 1 AND PERSON.SEX = 1 AND PERSON.AGE <= 20
ASSIGN 1
INCASE PERSON.RELAT = 1 AND PERSON.SEX = 2 AND PERSON.AGE <= 20
ASSIGN 2
INCASE PERSON.RELAT = 1 AND PERSON.SEX = 1 AND PERSON.AGE >= 21 and PERSON.AGE <= 45
ASSIGN 3
INCASE PERSON.RELAT = 1 AND PERSON.SEX = 2 AND PERSON.AGE >= 21 and PERSON.AGE <= 45
ASSIGN 4
INCASE PERSON.RELAT = 1 AND PERSON.SEX = 1 AND PERSON.AGE >= 46
ASSIGN 5
INCASE PERSON.RELAT = 1 AND PERSON.SEX = 2 AND PERSON.AGE >= 46
ASSIGN 6
RANGE 1 - 6
TYPE INTEGER
FOR HOUSIN.TYPHOU <= 8
VARLABEL "Households Head characteristics"
VALUELABELS 1 "Male under 20"
           2 "Female under 20"
           3 "Male between 21 - 45"
           4 "Female between 21 and 45"
           5 "Male over 46"
           6 "Female over 46"
```

The Output table gives us the following results:

Frequency				
of Households Head characteristics				
	Categories	Counts	%	Cumul %
	Male under 20	200	1.8%	1.8%
	Female under 20	61	0.5%	2.3%
	Male between 21 - 45	4,598	40.4%	42.7%
	Female between 21 and 45	888	7.8%	50.5%
	Male over 46	4,177	36.7%	87.2%
	Female over 46	1,451	12.8%	100.0%
	Total	11,375	100.0%	100.0%
	NotApp :	2,544		

The value for the new variable HOUSIN.AGE_SEX will depend on the value of the variables PERSON.SEX and PERSON.AGE for the household head: if SEX = 1 it means the household head is male and if AGE is less than 20 it means that the household is assigned the value of 1 for the new variable HOUSIN.AGE_SEX.

Notice that the collective houses are left aside with the expression FOR and only the private Households are considered.

- b) Another example is the case where you want to define a Household according to its unmet basic needs based on accessibility to services such as water, electricity or sewerage.

```
DEFINE HOUSIN.UBN
AS SWITCH
INCASE HOUSIN.ROOF = 5 OR HOUSIN.ROOF = 6
ASSIGN 3
INCASE HOUSIN.FLOOR = 7
ASSIGN 3
INCASE HOUSIN.WATER = 2
ASSIGN 2
OPTIONS DEFAULT 1
TYPE INTEGER
RANGE 1 – 3
```

The value for UBN will be 3 if ROOF = 5 or 6 or FLOOR = 7. Then the system checks the WATER variable, and if it is equal to 2, UBN will be 2, otherwise it will be assigned 1 (OPTIONS DEFAULT 1). Note that this condition works like an ELSE for the cases left outside the previous conditions.

Notes

- The SWITCH clause should be used when the new variable depends on more than one source variable.
- It works like an IF ... THEN ... ELSE, or a CASE structure of the regular programming languages.
- The testing and assigning conditions work in pairs, a testing condition should always be followed by an assigning expression.
- As soon as a testing condition (INCASE) is True, the system assigns the value given by the next ASSIGN command to the new variable and EXITS the evaluations. This is very important because the new value will depend on the order in which the conditions are written.

XIII. Using logical expressions in arithmetic expression

A logical expression is generally used in the context of a filter, i.e. a clause FOR but it can also be used as a condition to define a new variable under the AS clause. The result of a logical expression is of BOOLEAN type: TRUE or FALSE. The Red7language extends the BOOLEAN type to INTEGER so that arithmetic expressions may include logical expression with the following convention:

A TRUE expression takes the value 1
A FALSE expression takes the value 0

Under this convention the arithmetic expression:

(PERSON.AGE > 15)

takes the value 1 for any person of 16 years old and over and the value 0 for persons less than 16 years old.

Demonstration

Classify the households according to its type of walls: satisfactory (0) or unsatisfactory (1).

```
RUNDEF CMD23  
SELECTION ALL
```

```
DEFINE HOUSIN.UBNfloor  
AS (HOUSIN.WALLS = 4 OR HOUSIN.WALLS = 5)  
TYPE INTEGER  
RANGE 0 - 1
```

```
TABLE one  
AS FREQUENCY  
OF HOUSIN.UBNfloor
```

Exercises

1. Classify the household according to the floor of the house into two categories: 0 "satisfactory" and 1 "unsatisfactory".
2. Classify the household according to the type of toilet connection into two categories: 0 "satisfactory" and 1 "unsatisfactory".
3. Classify the household into two categories depending on the water origin: 0 "satisfactory" and 1 "unsatisfactory".

QUIZ 3

Name: _____

Date: _____

Please answer the following questions

1. When and why do we use the RECODE clause?

2. Identify the generations inside a household using the relation to household head using the following scheme:

<u>Generation of</u>	<u>Recode value</u>
Household head & spouse	1
Children	2
Grandchildren	3
Parents	4
Non identifiable (other)	6

3. When and why do we use the SWITCH clause?

4. Write down the command set using switch with the following requirement: Assign 1 to young females (10 – 17) attending school, assign 2 to young females (10 – 17) not attending school, assign 3 to young males (10 – 17) attending school, assign 4 to young males (10 – 17) not attending school.

XIV. Creating new indicators

Creating a new variable that defines the household Type (HHOLDTYPE)

Definition

A household type category is a way to differentiate the inner household family structure. It is a computed variable that has been processed with census variables extracted from the database.

Categories

1. Head of the household and spouse
2. Single headed household male without children
3. Single headed household female without children
4. Single female headed household with children < 15
5. Single male headed household with children < 15

Notes

The use of this variable helps to segment or stratify the population into groups. One of these groups is the “Single headed female household with children” which might be used as a pivot to compare basic household statistics such as water and toilet conditions, electricity supply, education level and any other variable that can provide insight on the population’s socio economic profile classified by family structure.

Promoting information – Selecting a unique record

Objective

The objective of this exercise is to demonstrate the technique for creating a new entity variable (age of household head) from a member entity variable (age of individual). This operation “promotes” a value from a lower level of the hierarchy. In various situations, a specific record of a lower entity may be promoted to its parent entity without losing any relevance. This is the case when the record has a unique characteristic within the entity children.

The traditional case is the individual of the household selected as the household head. In this case, any attribute of the person (age, sex, etc.) in some way also characterizes the household itself.

Based on this statement, a household can be characterized by the sex of its head. The variable at person level can be “promoted” to household level. The scheme is implemented with a FOR clause.

Define the “**sex of household head**” as the “**sex of the person**” who is “**identified as head of household**”. In other words, define at the “household entity level” a new variable called “sex of household” as the “sex of the individual” considered “the head of household”.

Translating this into Red7syntax and defining age and sex of the household head:

```
DEFINE HOUSIN.SEX
  AS PERSON.SEX
  FOR PERSON.RELAT = 1
```

```
DEFINE HOUSIN.AGE
  AS PERSON.AGE
  FOR PERSON.RELAT = 1
```

DEFINE indicates that a new variable is to be created. “HOUSIN.SEX” is the syntax's convention to define the variable identifier. The first part of the identifier is the entity name to which the variable belongs, then a dot, then the variable name. This name is user defined. The variable “SEX” belongs to, or “is member of”, the entity “HOUSIN”. The AS clause describes what value the new variable will take and the FOR clause is used to select the individual record. The LIKE clause can be used as a short cut to assign all attributes of a variable (type, range, size, etc.) to the new variable being defined. LIKE will assign all attributes other than those explicitly defined for the new variable (such as VARLABEL in the example below).

```
DEFINE HOUSIN.SEX AS PERSON.SEX
  FOR PERSON.RELAT = 1
  VARLABEL "Sex of Head of Household"
  LIKE PERSON.SEX
  SAVE "C:\Program Files\Redatam\NMIR\BASER\HOUSIN_SEX.rbf" OVERWRITE
```

Notes

1. The scheme is relevant when there is one and only one person defined as the head of household. Unless the data contains some errors this assumption is reasonable.
2. Whenever the previous assumption is FALSE the variable is considered MISSING. To prevent this situation a default value must be defined.

Deriving poverty indicators from the census Data

Objective

This module is focused on substantive matters more than in techniques. The selection of the appropriate variables is important to generate a profile of the households through various indicators. The Unmet basic needs (UBN) indicator measures the physical conditions under which population lives as well as their level of economic vulnerability.

Unmet Basic Needs

- Physical : Dwelling state
- Services and Utilities : Water, Electricity, Sewage

Vulnerability

- Household Head Education Level
- Dependency

A. Defining basic needs related to physical state of dwelling

Produce tables related to physical variables included in the household questionnaire. Comparison is made according to urban versus rural location and the year of building.

*Type of dwelling

```
TABLE table1
      AS FREQUENCY
      OF HOUSIN.TOILCO
      AREABREAK DISTRICT
```

*Ownership

```
TABLE table2
      AS FREQUENCY
      OF HOUSIN.WATERO
      AREABREAK DISTRICT
```

*Wall material

```
TABLE table3
      AS FREQUENCY
      OF HOUSIN.WALLS
      AREABREAK DISTRICT
```

*Roof material

```
TABLE table4
      AS FREQUENCY
      OF HOUSIN.ROOF
      AREABREAK DISTRICT
```

Note:

The AS clause is a mix between arithmetic and logical expression. A logical expression is always of type BOOLEAN (true, false). A BOOLEAN variable is NOT an INTEGER variable. However, a BOOLEAN value

True is coded as a 1 value and a False value is coded 0. The AS clause uses this unorthodox feature. The SWITCH command can be used also.

Defining a Sex & Dependency ratio

Among the traditional demographic indicators, dependency ratios and the sex ratio are commonly calculated. Red7 offers a very simple way to compute these indicators through the use of command sets.

Demonstration

a) Economic dependency is defined by measuring the ratio between the number of economically inactive people and the number of economically active people in a given area. The definition of active and inactive depends on the user definition and is generally calculated from an Economic Activity variable from the database.

Computation scheme

```
RUNDEF IND1
  SELECTION all

DEFINE PERSON.active
  AS RECODE PERSON.ACTYPE (1, 2 = 1) (3 – 9 = 0)
  TYPE INTEGER

DEFINE DISTRICT.active
  AS COUNT PERSON
  FOR PERSON.active = 1
  TYPE INTEGER

DEFINE DISTRICT.allpersons
  AS COUNT PERSON
  FOR PERSON.active = 1 OR PERSON.active = 0
  TYPE INTEGER

DEFINE DISTRICT.indic
  AS (DISTRICT.allpersons - DISTRICT.active) *100 / DISTRICT.active
  FOR DISTRICT.active > 0
  TYPE REAL

TABLE indicator
  AS AREALIST
  OF DISTRICT, DISTRICT.NDISTRI, DISTRICT.indic
```

Notes

1. Dividing by zero. The FOR clause prevents the program aborting if the number of actives in an area is equal to zero. In this case, the dependency ratio for the area is considered NOT APPLICABLE

b) The sex ratio of a given area is defined as the ratio of the number of males divided by the number of females in the area. The index is generally computed as a percentage from 0 to 100.

Command set

```
RUNDEF IND4
  SELECTION ALL

DEFINE DISTRICT.totmales
  AS COUNT PERSON
  FOR PERSON.SEX = 1
  TYPE INTEGER

DEFINE DISTRICT.totfemales
  AS COUNT PERSON
  FOR PERSON.SEX = 2
  TYPE INTEGER

DEFINE DISTRICT.indic
  AS 100 * ( DISTRICT.totmales / DISTRICT.totfemales )
  FOR DISTRICT.totfemales > 0
  TYPE REAL

TABLE DISTRICT.ratio
  AS AREALIST
  OF DISTRICT, DISTRICT.NDISTRI, DISTRICT.indic
```

Exercise

1. Compute a sex ratio for population between 15 and 45 at the County level.

Estimating the Education Level of the Head of the Household

Define for each household the level of education given by the education attained by the Head of the household. Define what you will consider as a low level and high level of education and then assign it to each person. Then pick the head of the household and promote it to the household itself.

Command set

```
RUNDEF IND4
  SELECTION all
  UNIVERSE PERSON.RELAT = 1

DEFINE PERSON.var
  AS RECODE PERSON.EDTYPE (0,2,3,4,5,6,7,8, 9 = 1) (1 = 0)
  FOR PERSON.AGE >= 15
  TYPE INTEGER

DEFINE DISTRICT.selection
  AS COUNT PERSON
  FOR PERSON.var = 1
  TYPE INTEGER

DEFINE DISTRICT.totrecords
  AS COUNT PERSON
  FOR PERSON.var = 1 OR PERSON.var = 0
  TYPE INTEGER

DEFINE DISTRICT.indic
  AS DISTRICT.selection *100 / DISTRICT.totrecords
  FOR DISTRICT.totrecords > 0
  TYPE REAL
  VARLABEL "Primary education only"

TABLE indicator
  AS AREALIST
  OF DISTRICT, DISTRICT.NDISTRI, DISTRICT.indic
```

Poverty indicator:

Socio-economic status of teenage mothers. Is poverty a determinant of teenage fertility?

Objectives:

- Acquire the capability to translate a conceptual problem into data requirements and command sets for statistical processing.
- Demonstrate the step by step analysis of a complex statistical process.

Strategy:

Definition of a target group; Cross tabulation with the target group and the total population; Analysis of differences between the target group and the total population. Consider the following example:

Step 1 - Target group definition

Group definition

Our population to target involves women up to 19 years old that have children that are alive.

To find these women we have several variables: age (AGE), sex (SEX), children that are alive (CHILDA) and live births (NCHILD). Using these variables (not necessarily all of them) we classify the persons that belong to this group saving the classification in a new variable (targetgr). The command set would look like the following:

```
RUNDEF Target
  SELECTION ALL
  DEFINE PERSON.targetgr
  AS SWITCH
  INCASE PERSON.SEX=2 AND PERSON.CHILDA > 0 AND PERSON.CHILDA <= 14
    AND PERSON.AGE < 20
  ASSIGN 1
  INCASE PERSON.SEX=2 AND PERSON.AGE < 20 AND (PERSON.CHILDA = 0 OR
    PERSON.CHILDA = 98)
  ASSIGN 2
  RANGE 1-2
  OPTIONS DEFAULT 0 NOTAPPLICABLE 9
  TYPE INTEGER
  VARLABEL "Women < 20 Target Group"
  VALUELABELS 1 "with kids" 2 "no kids"
  SAVE "C:\Program Files\Redatam\NMIR\BASER\targetgr.rbf" OVERWRITE
```

Step 2 - Household conditions using RECODE

The following step involves the creation of a more complex indicator that classifies the dwelling according to its conditions.

Analyze the following Red7command set:

```
RUNDEF IND2
  SELECTION all
  DEFINE HOUSIN.var1
    AS RECODE HOUSIN.WATERO (3 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var2
    AS RECODE HOUSIN.FLOOR (7 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var3
    AS RECODE HOUSIN.WATERO (3 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var4
    AS RECODE HOUSIN.ROOF (5,6 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var5
    AS RECODE HOUSIN.STLGHT (2 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var6
    AS RECODE HOUSIN.WALLS (4,5 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var7
    AS RECODE HOUSIN.TOILCO (2 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var8
    AS RECODE HOUSIN.KITCH (2 = 1) ELSE 0
    RANGE 0 : 1
    TYPE INTEGER
  DEFINE HOUSIN.var
    AS HOUSIN.var1 + HOUSIN.var2 + HOUSIN.var3 + HOUSIN.var4 +
      HOUSIN.var5 + HOUSIN.var6 + HOUSIN.var7 + HOUSIN.var8
    RANGE 0-8
    TYPE INTEGER
    VARLABEL "Households with UBN"
    VALUELABELS 0 "none" 1 "1 UBN" 2 "2 UBN" 3 "3 UBN" 4 "4 UBN"
                5 "5 UBN" 6 "6 UBN" 7 "7 UBN" 8 "8 UBN"
  DEFINE DISTRICT.indic
    AS COUNT HOUSIN
    FOR HOUSIN.var >= 3
    VARLABEL "Number of Households w/more 3 ubn by ED"
    TYPE INTEGER

  TABLE DISTRICT.ratio
    AS AREALIST
    OF DISTRICT, DISTRICT.NDISTRI, DISTRICT.indic
```

Now, it is necessary to modify this indicator to classify each household, by means of a new household variable, instead of just counting them for each DISTRICT. To do this modify the last DEFINE command

```
DEFINE HOUSIN.indic
  AS HOUSIN.var >= 3
  RANGE 0-1
  VARLABEL "Household w/more 3 ubn"
  VALUELABELS 0 "< 3 UBN" 1 ">= 3 UBN"
  TYPE INTEGER
```

Step 3 - Household dependency ratio

Dependency ratio at household level.

The dependency ratio measures the burden on the active persons in the household who have to maintain the inactive persons. The active members in the household can be obtained by the variable Type of Activity where we find the persons that worked, look for a work, studied, etc. during the week previous to the census. From this variable we get the active members and from the count of all the members in the household we obtain the indicator. There are two ways of computing a dependency ratio: a) where we measure the percentage of active members in the household, and b) where we obtain the number of dependents by each active member.

- a) computes % of active persons in the household

```
DEFINE HOUSIN.actives
  AS COUNT PERSON
  FOR PERSON.ACTYPE = 1 OR PERSON.ACTYPE = 2
  TYPE INTEGER
```

```
DEFINE HOUSIN.totpers
  AS COUNT PERSON
  TYPE INTEGER
```

```
DEFINE HOUSIN.depratio
  AS ( 100 * HOUSIN.actives / HOUSIN.totpers )
  FOR HOUSIN.totpers > 0
  TYPE INTEGER
  RANGE 0-100
```

- b) computes the number of dependents by each active person in the household

```
DEFINE HOUSIN.indic
  AS (HOUSIN.totpers - HOUSIN.actives) / HOUSIN.actives
  FOR HOUSIN.actives > 0
  TYPE REAL
  OPTIONS DEFAULT 0
```

Adding a threshold or a break point to this count allows us to classify the household according to whether it has a high or low dependency ratio.

Assignment:

Classify those households with more than or less than 30 % of active members. What is the distribution of the dependency ratio?

Compare this indicator with our target group. What is the percentage of teenage mothers that live in households with less than 30 % of active members?

Step 4 – Education Dimension

Education is measured in several variables so we have to define which ones are going to be analyzed. For example: Type of Education (EDTYPE), School Attendance (ATTEND), Last Course Approved (i.e. passed) (COURSE).

Assignment:

1. Calculate the percentage of teenage mothers (less 20 years old) that only reached primary level.
2. What is the relationship between teenage motherhood and education level? Is there any difference with the rest of the population?
3. Calculate which of these teenagers gave birth. Has the burden of childbearing limited their education?

Step 5 – Working situation

Assignment:

1. Analyze the economic situation of our target group compared with the population in general.
2. Focus on the total population and extract the characteristics of heads of household such as education level, economic activity, unemployment, etc. and analyze the distribution.
3. From here get the female heads of household only.
4. Determine the age of the heads of the household to compare the situation of young heads of household by sex.

Step 6 - Indicator summary

After you have analyzed the different characteristics of teenage mothers including their family structure, education, level, economic activity, vulnerability and poverty you can summarize the analysis presenting a full report with tables, outputs, and maps.

Matrix Operations example:

The MATRIX operation is a function that operates over computed tables that have the same dimension (same number of rows and columns). The operations are very similar to the ones of common matrices (multiplication, division, sum, subtractions). It is very useful when you need to get percentages and ratios of values ordered in table format.

EXAMPLE: Compute a dependency ratio based on age

```
RUNDEF job
  SELECTION ALL
```

*****1. CREATE THE VARIABLE AREA AT THE PERSON LEVEL *******

```
DEFINE PERSON.AREA
  AS VAL (AREA.AREA)
  RANGE 1-2
  TYPE INTEGER
  VALUELABELS 1 "URBAN" 2 "RURAL"
```

*******2. CROSSTABULATION OF URBAN-RURAL BY SEX BUT JUST FOR PEOPLE LESS THAN 14 AND
*** OVER 65 YEARS OLD *******

```
TABLE P1
  TITLE "PEOPLE LESS THEN 14 AND OVER 65"
  AS CROSSTABS OF PERSON.AREA BY PERSON.SEX
  FOR PERSON.AGE <=14 OR PERSON.AGE >= 65
```

****** 3. CROSSTABULATION OF URBAN-RURAL BY SEX BUT JUST FOR PEOPLE BETWEEN 14 AND
*** 65 YEARS OLD *******

```
TABLE P2
  TITLE "PEOPLE BETWEEN 14 AND 65"
  AS CROSSTABS OF PERSON.AREA BY PERSON.SEX
  FOR PERSON.AGE >= 15 AND PERSON.AGE <= 64
```

******* 4. TO GET THE DEPENDENCY INDICATOR (BASED ON AGE) YOU DIVIDE TABLE 1 BY TABLE 2**

```
TABLE P3 AS MATRIXOP OF P1, DIVISION, P2
  OPTIONS  DECIMALS 3
```

***** 5. MULTIPLY THE OUTPUT BY 100 TO GET THE PERCENTAGE ****
**** IT MEANS HOW MANY CHILDREN AND ELDERLY DEPEND ON THE REST OF PEOPLE**

```
TABLE Tab1
  TITLE "DEPENDENCY RATIO BASED ON AGE"
  AS MATRIXOP
  OF P3, MULTIPLICATION, 100
  OPTIONS  DECIMALS 1
  OUTPUTFILE XLS "C:\.....\Tab1.XLS"
  OVERWRITE
```

Defining a map composition

This Red7 map version (Beta version) allows you to display thematic maps of database variables that are associated with a geographical code. This feature is very important for planning, since the maps show at first site which areas have a similar value, and which areas are different, according to the mapped variable. By using different cartographic coverages you can see how the geographic objects such as rivers, mountains, etc., can affect the behavior of the variables being displayed. (For example, the access to schools can be relevant when explaining different literacy levels).

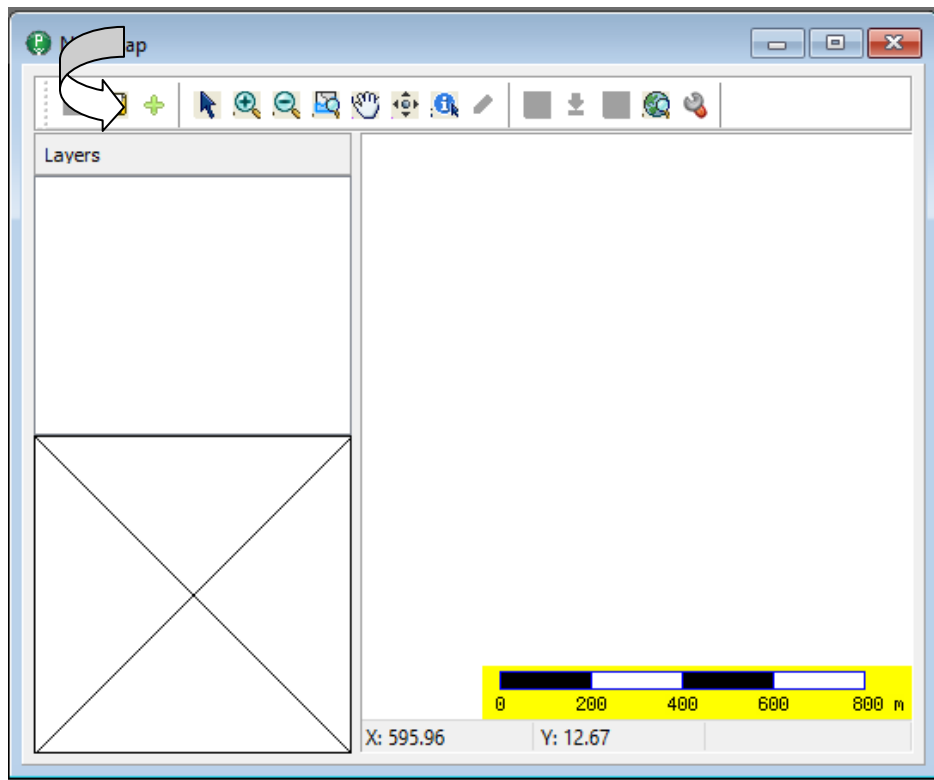
The requirement to display a map in Red7 is to connect the Redatam database with a digital map which must be in a specific format known as shape files (shp) from the ArcView™ mapping tool. Redatam DOES NOT build digital maps, it only converts the SHAPE digital map to an internal Red7working format.

Once the database is opened, the procedure to connect a map to the Redatam database is executed by calling the Map Composition window.

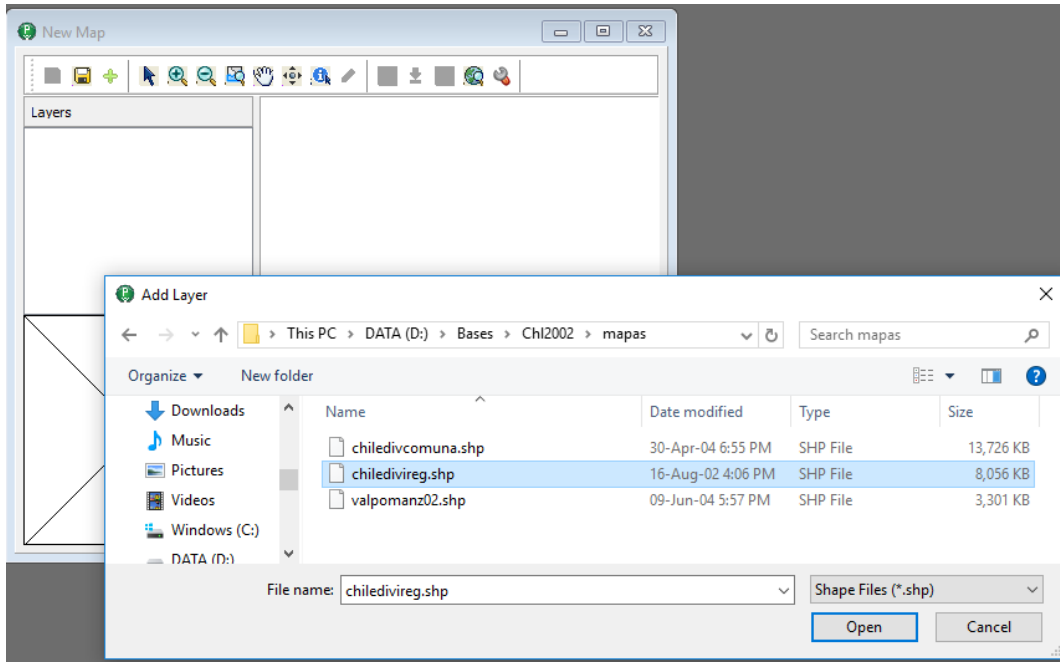
XV. Procedure

In the main menu, choose the option File >New >Map Composition.

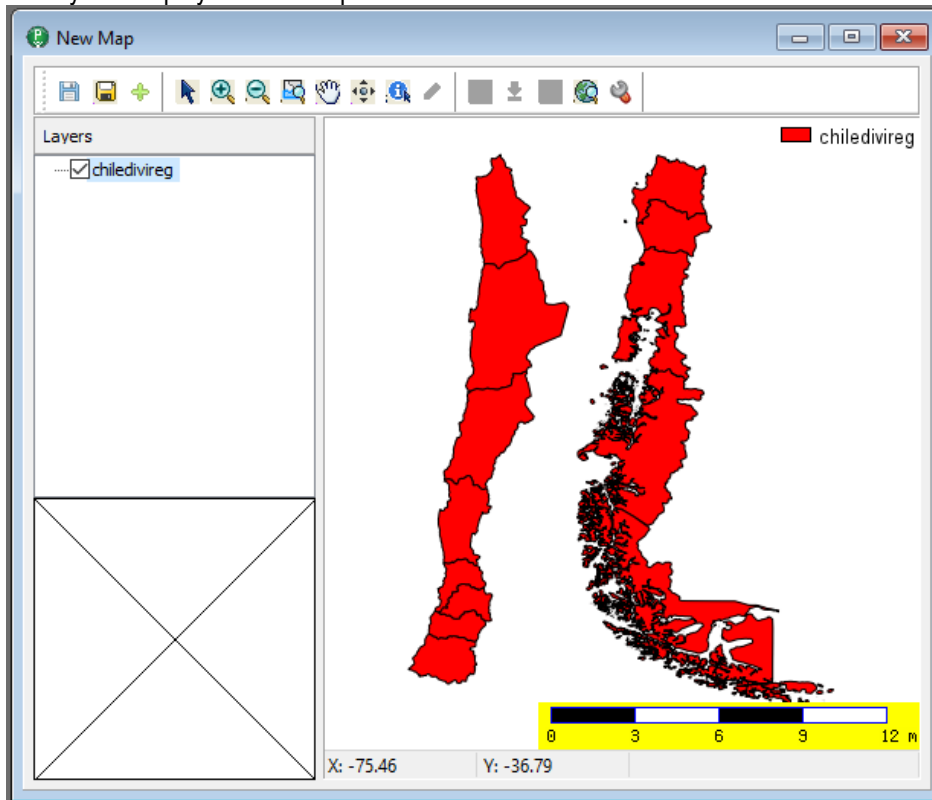
The system will show the Map window. Click the arrow icon from the tool bar. Red7will open a new dialog box to allow you to choose a Shape file (generated by ArcView). The file must contain the same entity elements as exist in the database.



Select the *.shp file and press the **Open** button to accept it.

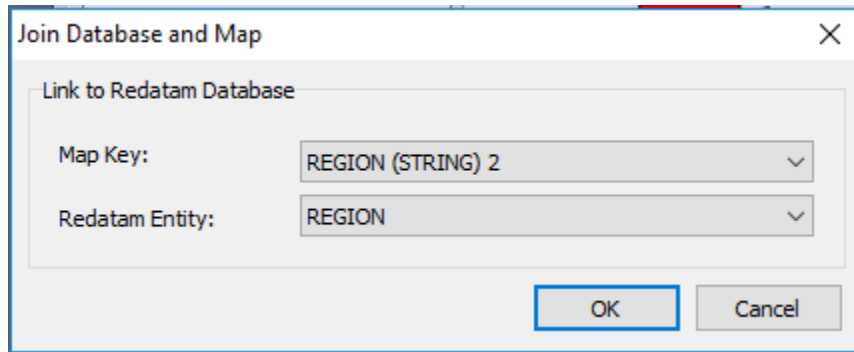


The layer is displayed in the map window.



First of all, it is necessary to join the shape file with the database. With the right click of the mouse open the popup menu and select the **Redatam Config** option. Fill the upper combo box with the field of the shape file

containing the key field and the lower combo box with the proper entity for this shape file (i.e. REGION), press the OK button.



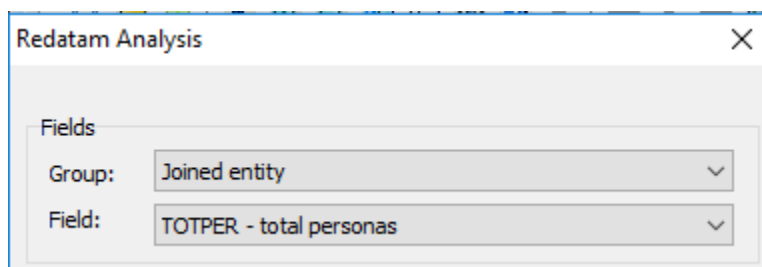
The **Join** is the **most important action with the map**, it is used to associate the map with the database. The system shows a combo box with all the variables in the shape file. You should select the field from the map that holds the key field (Redatam code) associated with the Redatam database (related to the entity level that the map is representing), for example a district map is related to the district code in the Redatam database. Remember that the code in Redatam is a compound code. Select the entity level in the lower combo box.

Save the Map Composition Properties

Once you have defined all the properties of the Map Composition, you should save it by using the **Save** button in the tool bar. It will be saved in a text file with an .mxd extension, and this file can be edited with any text editor if you want to change its contents.

This Beta version of the mapping window only accepts variables that are already saved at the entity level of the shape file (in this example REGION). So in order to map a variable this variable must exist in the database.

With the right click of the mouse open the popup menu and select the **Redatam Analysis** option.



First select the group where to get the variables to map, this should be Joined entity. And then select the variable to map (i.e. TOTPER). Once an indicator has been created, the results are displayed in the output table and the variable name appear in the dictionary window at the same time.






Then, select the type of classification (i.e., quintiles), the color palette and the number of categories you want to depict. Then press the DO button in order to get each class with its proper color.

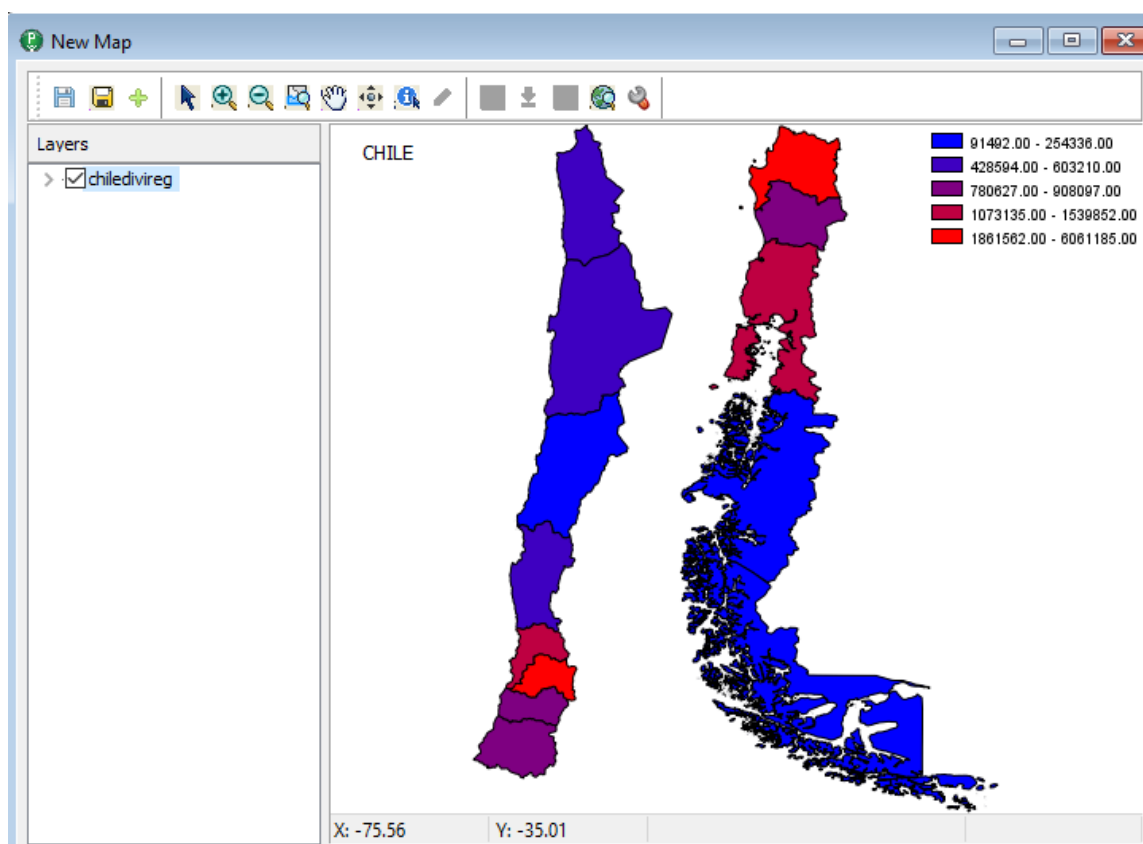
Classification

Classification:

Colors:

Classes:

Symbol	Range
	91492.00 - 254336.00
	428594.00 - 603210.00
	780627.00 - 908097.00
	1073135.00 - 1539852.00
	1861562.00 - 6061185.00



The map was produced based on the variable TOTPER created at the REGION level.

The map properties such as the Title, Subtitle, Legend, Coverage and Classifications were defined in the map properties window opened with the tools icon from the tool bar.

