BASICS OF \textit{winR+}©

REDATAM-Plus for Windows vers. 1.1
(REtrieval of DATa for small Areas by Microcomputer)

Software developed by CELADE in the context of a joint CELADE-University of Waterloo Project financed by the International Development Research Center (IDRC) of Canada with additional support to CELADE from the United Nations Fund for Population (UNFPA), the United Nations Regular Budget, the Canadian International Development Agency (CIDA) and the Inter-American Development Bank (IDB).

UNITED NATIONS
ECONOMIC COMMISSION FOR LATIN AMERICA AND THE CARIBBEAN (ECLAC)

LATIN AMERICAN DEMOGRAPHIC CENTRE (CELADE)
Casilla 91 - Santiago, Chile
Tel: (562) 210-2000  Fax: (562) 208-0252
SUMMARY OF CONTENTS

Detailed Table of Contents .................................................. iv

Preface

and

Acknowledgments ................................................................. xi

Chapter 1.

WHAT IS winR+? ............................................................... 1

Chapter 2.

INSTALLING winR+ ........................................................... 13

Chapter 3.

BASIC winR+ CONCEPTS ..................................................... 21

Chapter 4.


Chapter 5.

A WALK THRU winR+: SPEAKING THE LANGUAGE [Tutorials 3-4] .... 57

Chapter 6.

COMMAND SETS: FURTHER LEARNING THRU EXAMPLES ............ 109

Chapter 7.

ADDITIONAL TOPICS and PROCEDURES ................................. 137

Appendix A.

WHAT IS NEW FOR REDATAM-PLUS USERS ............................... 203

Appendix B.

REFERENCE GUIDE TO THE winR+ COMMAND LANGUAGE .......... 217

Appendix C.

GLOSSARY ............................................................................. 267

INDEX .................................................................................... 273
# Detailed Table of Contents

**Preface**

Acknowledgments ................................................. xi
How to use this manual ......................................... xii
The Reference Guide to the Command Language ............... xiv
Accepting change: the Manual vs. the Software ............... xiv
Conventions used in the manual ................................ xv
  Symbols xv
  Figures xv
  Typographical conventions xvi
  Speaking of Households and Houses xvii

Acknowledgments ................................................. xviii
How to obtain more information ................................. xxi

*Chapter 1.*

WHAT IS *winR*+? .................................................. 1
  Brief Summary of *winR*+ ................................... 1
  Major Features of *winR*+ .................................. 2
    Workspace for database and user information 2
    Geographical selection 3
    The statistical processor language 5
    An open system: *winR*+ and external databases 6
    Hierarchical processing 6
    Outputs: Frequencies, Cross-tabulations, Averages and AreaLists 7
    Speed of processing 7
    The *winR*+ Data Link: GIS and more 7
    Multilingual facilities 8
    Data security 8
    The *winR*+GIS decision support system tools 9
    The *winR*+ building blocks 9

Where Does the Database Come From? ............................ 10
Who are the Intended Users of *winR*+? .......................... 10
  End-users 10
  Database administrators 11
  Builders of *winR*+GIS tools and other applications 11
Chapter 2.
INSTALLING winR+ ........................................... 13
  Requirements for using winR+ ............................... 13
    User knowledge ........................................ 13
  System requirements ...................................... 13
    Recommended configuration 13
    Minimum configuration 14
Installing winR+ ........................................... 15
  Installing winR+ in Windows 3.x 15
  Installing winR+ in Windows 95 15
Launching winR+ ............................................ 16
Customization .................................................. 16
The New Miranda demonstration dataset ...................... 19

Chapter 3.
BASIC winR+ CONCEPTS ...................................... 21
  winR+ database concepts ................................. 21
    Entities and Elements 21
    Selectable and unselectable elements 23
    Variables 24
Some definitions for working with winR+ .................... 24
  Dictionary 24
  Database structure 24
  Selection set 24
  Command set 25
  Tables: Frequencies, Crosstabs, Averages and AreaLists 26
  AreaLists 26
  Run 26
  Workspace 26
  Assist 27

Chapter 4.
  TUTORIAL-1: Navigating in winR+ ......................... 30
    The winR+ main screen .............................. 30
    The Icon (Tool) Bar and its icons .................. 32
Basics of winR+

Living in a Workspace ............................................. 33
   The Workspace window 33
   Viewing the dictionary and database structure 36
Obtaining Help in winR+ ............................................. 37

TUTORIAL-2: Simple information retrieval .......................... 39
   Creating a Selection Set ........................................... 39
   Creating a simple Selection Set 40
   Creating a Selection Set in a multi-branch database 42
   An alternative method: mouse clicking a Selection Set 43
Writing commands with a mouse: EasyTabs .......................... 44
   When EasyTabs can be used 45
   Frequency distributions with EasyTabs 45
   Cross-Tabulations with EasyTabs 48
   Averages with EasyTabs 51
Counting things in sub-areas with Simple Count .................... 51
Viewing results via the Output Report ............................... 54

Chapter 5.
A WALK THRU winR+: SPEAKING THE LANGUAGE [Tutorials 3-4] .......................... 57
TUTORIAL-3: The winR+ language ................................. 58
   Identification of variables in winR+ ........................... 58
   Introduction to the winR+ command language [Tutorial 3a] .... 59
      The three basic commands: RUNDEF, DEFINE & TABLE 59
      Assembling commands and keywords in a Command Set 60
      Using the Reference Guide to the Command Keywords 62
      Getting an Assist 62
      A simple CmdSet with RUNDEF and TABLE [Tutorial 3a] 62
      Excluding ("filtering") cases with UNIVERSE and FOR 72
      AREABREAK to repeat an output for each sub-area 75
      Processing the entire database: Selection Set ALL 77
Defining new variables [Tutorial 3b] ................................ 78
   Working with the General Assist 78
   Grouping categories with DEFINE...AS RECODE 82
   Filtering a DEFINE with FOR 83
   Naming with VARLABEL and VALUELABELS 84
   Stating the RANGE and TYPE of a new variable 85
   Using the new variables in TABLES 86
Basics of \textit{winR+} — vii

Defining a new variable at a higher entity level [Tutorial 3c] \hfill 87
CROSSTABS of variables at different entity levels [Tutorial 3d] \hfill 92
Saving a TABLE as a spreadsheet for post-\textit{winR+} manipulation \hfill 94

\textbf{TUTORIAL-4: The \textit{winR+} language for AREALISTS} \hfill 95
Aggregated variables for an AREALIST [Tutorial 4a] \hfill 96
- More on counting elements in each of many areas 96
Creating an AREALIST [Tutorial 4b] \hfill 99
- Aggregate variables at the AREALIST entity level 99
Becoming familiar with the AREALIST Browser 101
Variables below the AREALIST entity level: Frequencies 103
DBF and other filetype alternatives for an AREALIST 106
Exporting an AREALIST to a spreadsheet package 107

\textbf{Chapter 6.}
\textbf{COMMAND SETS: FURTHER LEARNING THRU EXAMPLES} \hfill 109

Five-year age groups by a formula \hfill 110
Related topics \hfill 112
- \textit{winR+} operators 112

Number of generations in a household \hfill 113
Related topics \hfill 115
- DEFAULT values vs. NOT APPLICABLE values 115
LIKE for writing less 115
Hierarchical processing limitation: Move up, Stay up 116
Care of bugs in your base 116

Average age of persons in a household \hfill 117
Related topics \hfill 118
Use of integer division 118

The age of the oldest person in a household \hfill 119
Related topics \hfill 120
- The age but not the person 120
The sex of the oldest-aged person in a household \hfill 121
The age of the youngest person in a household \hfill 123
Household overcrowding \hfill 124
Related topics \hfill 126
- Boolean variables 126
- Concern for data quality 126

\textbf{WEIGHTing TABLEs} \hfill 126
Combining variables to create a new variable \hfill 128
The number of items in a household \hfill 130
Sequential numbering of elements in an entity ............................................. 132
Identifying areas with sequences of adjacent houses with the same characteristics 133
  Related topics ....................................................................................... 136
  Creating a Selection Set from an AREALIST 136

Chapter 7.
ADDITIONAL TOPICS and PROCEDURES .................................................. 137
Database conversion & Workspace management ........................................... 138
  Converting a Redatam-Plus database to winR+ ..................................... 138
  Creating a Workspace for an existing winR+ database ............................ 140
  Importing Workspaces, databases, documents ........................................ 140
    Initialization of an imported Workspace and/or database 141
  Import and Export of Command Sets ...................................................... 142
Using external database variables .............................................................. 143
  External variables in a CmdSet: DEFINE...AS DATASET ....................... 144
  Adding external variables to your Workspace dictionary ....................... 148
Saving DEFINEd variables in a winR+ database ........................................ 153
Extending hierarchical processing in winR+ ............................................. 156
  Quasi pre-processing ........................................................................... 156
    Using TOTCASES for quasi pre-processing 157
COUNT vs. TOTCASES ............................................................................ 159
  Virtual multiple-pass processing SAVE variable ................................... 161
Working in the AREALIST Browser ............................................................ 162
  Overview of the AREALIST Browser ...................................................... 163
    Records 163
    Columns 165
Importing and Exporting AREALISTS ....................................................... 167
Selecting and saving records of an AREALIST .......................................... 168
  The Query Expression Builder for record selection 168
  Saving the selected records in an AREALIST 171
Adding new columns (variables) to an AREALIST .................................... 173
Linking two AREALISTS ......................................................................... 176
Deriving a Selection Set from an AREALIST .............................................. 177
Creating Selection Sets using computed criteria ..................................... 177
  SelSet from semi-manual identification of specific areas 178
  Specific areas using the Query Expression Builder 180
Generating a SelSet directly in a CmdSet ............................................... 181
Using an external Selection File ............................................................... 182
Using AREALISTs with the winR+ map display .......................... 183
Adding a map definition to your Workspace .......................... 183
Linking an AREALIST to a map definition ........................... 185
Classifying a variable and displaying it on the map ..................... 187
  Classification of a variable 188
  Displaying selected records from an AREALIST on a map 192
  Editing the map objects and layout 193
Selecting and Querying areas on the AREALIST and map ............... 193
Creating a Selection Set by clicking areas on a map .................. 194
Moving beyond winR+ via the winR+ Data Link ......................... 196
Exporting an AREALIST in .dbf format to an external system ........ 197
Moving an AREALIST to an external target via winR+ Data Link ...... 198
  Detecting inconsistencies between the source and target 211

Appendix A.
WHAT IS NEW FOR REDATAM-PLUS USERS ....................... 217
  New features in winR+ .................................. 218
  Examples of Command Sets in winR+ and R+ ...................... 218

Appendix B.
REFERENCE GUIDE TO THE winR+ COMMAND LANGUAGE ............ 219
  Overview of the winR+ Command Language ....................... 220
    The three basic commands: RUNDEF, DEFINE & TABLE ............ 222
  Command structure and syntax summary .......................... 224
  Operators for expressions in Command Sets ...................... 227
Looking up Keywords in the Reference Guide ......................... 230
  Index to the Command Keywords ................................ 250
RUNDEF syntax and usage ...................................... 267
DEFINE syntax and use ....................................... 273
TABLE syntax and use ....................................... 249

Appendix C.
GLOSSARY ............................................... 267

INDEX ................................................ 273
Redatam-Plus for Windows, abbreviated winR+®, works with a very large compressed datasets that may have microdata describing each of millions of persons, houses, city blocks and other units from various censuses, surveys and other sources of an entire country. The winR+ software helps a user to select any geographical area of interest down to city blocks from the entire dataset, and then to define new variables and produce tabular outputs rapidly via graphical windows, all without the assistance of a programmer. winR+ allows data from different levels, such as persons, households, houses, blocks, districts and provinces, to be “hierarchically combined” according to the needs of the analyst, and, if desired, made available for map display or spatial analysis via Geographical Information Systems (GIS).

winR+ can be used with either Microsoft Windows 3.x or Windows 95 on all IBM-compatible microcomputers.

winR+ is the third generation of the Redatam software (Retrieval of Data for small Areas by Microcomputer). If you are a user of Redatam-Plus (for DOS), which is the second generation software, you will find the convenience of working in the Windows graphical environment, the many improvements and enhancements, as well as much increased speed and reliability, strong incentives for changing to winR+. You continue to use your old databases without change and only have to make an easy and
quick conversion of the dictionary. Since the basic concepts and operation are similar even though the user interface is totally new, users of Redatam-Plus will find it easy to move on to winR+.

This manual can be used for a) learning to use the system as a beginner, b) getting going fast as a previous Redatam-Plus user, or for c) learning about the software to decide whether to acquire it. The manual can also be used as a reference when working with winR+. The manual assumes that you are familiar with the Windows system on your computer and does not explain how to use Windows 3.x or 95.

How to use this manual

The contents of the manual are divided into seven chapters and three appendices:

1. What is winR+?
2. Installing winR+.
3. Basic winR+ concepts.
4. A walk through winR+: Becoming familiar [Tutorials 1-2].
5. A walk through winR+: Speaking the language [Tutorials 3-4].
7. Advanced topics and procedures.

Appendix A: What is new for Redatam-Plus users.
Appendix B: Reference Guide to the winR+ command language.
Appendix C: Glossary

How you use the manual, will depend largely on your purposes and background. Assuming that you are an analyst or other substantive data user, and depending on your purpose, we have suggested different approaches, one of which may suit your situation. Special cases, such as programmers or database administrators, or analysts very experienced in a variety of other software can get started in their own ways.

So that you have an immediate source of data with which to practice, and to avoid your having to create or obtain a winR+ dataset, the software
comes with a set of data for the hypothetical country of New Miranda. All examples in the Manual are based on this data.

The approaches suggested for using the manual are:

**PURPOSE: To learn to use winR+.
BACKGROUND: No previous Redatam-Plus experience**

Peruse the Preface and Chapters 1 and 3 to learn what winR+ is and the basic concepts. Then install the software using the Install and Launching winR+ sections of Chapter 2. Work through the Tutorials of Chapters 4 and 5.

Sometime after gaining experience with winR+, perhaps on your own data, you may want to glance through Chapter 6 to become familiar with some tricks of using the Command Language and read the section titles of Chapter 7 so you know other things that winR+ can do. Then, if and when you have a need to become more advanced in your use of the software, you can read specific sections of these chapters.

**PURPOSE: To learn to use winR+.
BACKGROUND: Experienced user of Redatam-Plus.**

After glancing through the rest of the Preface, look through Appendix A which outlines what is new in winR+ in comparison with Redatam-Plus. Then, after installing winR+ (Chapter 2), look through the Tutorials in Chapters 4 and 5. Since the command language is new, you may wish to work through some of the Command Set (CmdSet) examples in Chapter 5. Note that the second section of Appendix A has a number of the CmdSets written in both the winR+ and the R+ languages.

You then may want to go immediately to Chapter 6 to look at examples of more sophisticated Command Sets and also to peruse through Chapter 7 for topics of immediate interest to you.

When learning the winR+ language, you will find the *Reference Guide to the winR+ Command Language* in Appendix B useful for clarifying details that are not treated in the Tutorials.

**If you only want to make a few quick tabulations:** If your computer has a database of interest available and you need only a few quick tabulations, you may find it sufficient to use *EasyTabs*, which are covered in Tutorials 1 and 2 of Chapter 4. *EasyTabs* do not require any knowledge of the winR+ command language.
PURPOSE: To learn about winR+ for possible acquisition.
BACKGROUND: Little or no experience with Redatam-Plus.

Read Chapter 1 to learn what winR+ is and Chapter 3 to understand the basic concepts. To see what winR+ can do, look at the first paragraphs of each of the four Tutorials in Chapters 4 and 5.

The Reference Guide to the Command Language

Once the system is learned, all users will find the *Reference Guide to the winR+ Command Language* in Appendix B convenient to clarify specific points about the commands.

Accepting change: the Manual vs. the Software

The winR+ software, like many others, is constantly being enhanced and improved—and corrected when errors are found. These changes are incremental and the overall version number of winR+ stays the same—only the Rev[ision] number changes—until major enhancements are introduced, at which time, the overall version number will be changed. The About screen and other Help screens and the Readme file will reflect the changes.

The printed Manual, on the other hand, is static and cannot be reprinted frequently at reasonable cost. Hence, you may find various details, such as screen layouts or minor enhancements, that have changed or been added in the software since the manual was printed. The manual is based on winR+ [version 1.1].
Conventions used in the manual

Symbols

The Manual uses some conventions that inform the reader when a paragraph may be of special interest (or disinterest for specific users) or when special caution should be used. The following conventions are used in the left column facing the paragraph in question:

---

This indicates a recommendation or a “Tip” with additional information that will help you to better use or understand winR+, but which is optional and of less interest to persons who are just perusing this Manual to see if the software is of interest.

---

This symbol is a warning that you should exert particular CAUTION in working with winR+ at this point because the “road is slippery”, that is, it is easy to make a serious error that might have unfavorable consequences — usually not disastrous, but often requiring extra work to recuperate —.

---

Compared with R+

This indicates a comparison of winR+ with Redatam-Plus for DOS (abbreviated R+) and, while of special interest to persons familiar with R+, it will be of scarce interest to those who have never used R+.

Figures

Figures are numbered consecutively throughout the entire Manual. All diagrams, screen images and Command Sets (programs telling winR+ what to do) that are not part of a paragraph are included as Figures, making it easier to find a Figure. Note that many Figures have a caption that has a small cryptic name in curled brackets { } in the lower right hand corner of the caption; for example, Figure 2 on page 4 has \text{(risDB1.wpg)}. This should be ignored by the reader — it facilitates updating the manual in future editions and languages —.

With very few exceptions that are note in their captions, all Figures from winR+ were made in Windows 95; screens in Windows 3.x are essentially the same except for the Title Bar.
Typographical conventions

The following typographical conventions are used in the manual text:

**Italics**
Terms used for the first time, e.g., *entity element*.

Filenames; Workspace names; Selection Set names; Command Set names. E.g., Selection Set *Tutor2b*

Window names. E.g., *Selection Editor* window. The “Workspace”, which the user sees as a window, is not italicized since it is also a database file.

Software packages, such as *AccessPlan*, produced in the same collaborative project of CELADE with the University of Waterloo.

**Bold italics**
Entities, e.g., *90block, 90person*

**Bold**
Variable names, which are normally composed of the entity of the variable followed by the variable name, itself, with a period in between, e.g., *90person.90sex*

Instructions for the computer typed by the user, e.g., *a:setup*

Names of buttons, icons, etc., in windows (forms) within *winR+*, e.g., *Select/Deselect Button.*

Menu items to be entered in sequence, normally by the mouse clicks: *File/Open*, where the “|” indicates that *Open* is on a sub-menu of *File*.

**BOLD UPPERCASE**
Keywords of the *winR+* command language, e.g., *DEFINE, RECODE, AS, FOR* (note that *winR+* accepts commands written in Uppercase as well lowercase or any combination of these).

If the command has a set of required arguments to be supplied by the user, this is indicated in the following example:
OF <list of variables>

When the actual command is written in the text, this will appear as:

OF 90person.90age, 90housin.90water

UPPERCASE

Names of keyboard keys, e.g., ENTER or RETURN; CTRL (CONTROL). Note that if two keys are to be pressed at the same time, a “+” is used, e.g., CTRL+SHIFT.

When a Command Set, or part of it, is given in the manual, the following conventions are used:

Bold, non-Serif font

To set it off clearly, a Command Set (CmdSet) is written in a different font from the regular text of the manual, e.g.,

RUNDEF Tutorial_4b

SELECTION Tutor2b

BOLD, UPPERCASE

Keywords within the Command Set (see previous example). Typeface depends on whether it is in the text or in a full command.

More specific conventions will be explained as required.

**Speaking of Households and Houses**

The database used for examples throughout this Manual is based on census and other data of the hypothetical country of New Miranda. Since it is impossible to invent a reasonably consistent dataset of the level of complexity of the many datasets used in the New Miranda database, in fact, the data were taken from parts of one of the 150 or so real countries in the world; the names of places and the geography have been totally changed. The census data that forms the core of the multi-disciplinary New Miranda database distinguishes *houses* (or dwelling units) and within them, persons. Although some houses in the real world have more than one *household*, the census authorities “arranged” that there be only one *household* in each house.

As a convention in this manual, the text usually refers to the “household head”, the “number of persons in the household”, etc., since this is the normal convention even where most houses have only one household. In
New Miranda this is always the case, so the household always refers to the “house” as well.

Acknowledgments

Redatam-Plus for Windows, winR+, the third generation of the Redatam software package, was developed by CELADE within the joint R+GIS Population-Related Tools to Facilitate Decentralized Local Development Project of CELADE and the University of Waterloo, which was funded by the International Development Research Centre (IDRC) of Canada. The work in CELADE was also supported by the United Nations Fund for Population (UNFPA), the Inter-American Development Bank (IDB), the United Nations Regular Budget and the Canadian International Development Agency (CIDA).

Serge Poulard, Head of the CELADE Data Processing Unit, wrote the winR+ software, which is the most feature-laden and generalized version of Redatam. He totally re-designed the existing system to take advantage of Windows and advanced programming approaches, introduced new concepts to make an open and easier-to-use system, created the new winR+ command language, and wrote the software essentially single-handed in Microsoft Visual Basic®.

In the last stages, when Serge was optimizing the system to increase its processing speed, he had assistance for a number of weeks from Ari Silva. Ari, former Head of the CELADE Data Processing Unit, teamed up with Serge to re-program the winR+ statistical engine, which was originally prototyped by Serge in Visual Basic®, in Microsoft Visual C++®. The IBGE of Brazil made Ari available for these purposes under an agreement between the IBGE and CELADE. Within the joint CELADE-University of Waterloo Project, Serge also had the opportunity to spend a number of weeks at the University working with Professor Brent Hall and his team of Robert Bowerman, Gunnar Hillgartner and Robert Feick, during the early stages of planning winR+ and the development of the original R+GIS Link component of winR+ (now called the winR+ Data Link).

Alejandra Silva, a long-term CELADE GIS consultant, assisted Serge on the user-interface of ZonPlan, which he developed while working full-time on winR+. ZonPlan, a R+GIS front-end tool to assist planners that uses components of winR+ to create indicators and display them on maps
for identifying target populations, was applied in the Costa Rican Canton of Escazú by Alejandra who collected the necessary data and maps, digitized the information and trained the local Escazú staff. ZonPlan is one of the four R+GIS tools developed in cooperation with the University of Waterloo in the joint Project mentioned above. Alejandra also helped Serge on various aspects of winR+ related to geography and GIS, the user interface and the multilingual interface.

During the period of development, Claudio Meza, member of the CELADE Data Processing Unit, contributed his experience gained in the design and development of the previous generation, Redatam-Plus, and its utilization in the countries, maintained the DOS-based system and carried out a very large number of technical assistance missions to ensure that Redatam-Plus continued to be used widely in the Latin American and Caribbean region, thereby preparing these users for the introduction of winR+.

Arthur Conning, former Chief of the CELADE Population Information and Technology Area, developed the joint R+GIS Project with his counterpart, Professor Brent Hall, of the University of Waterloo, and supervised the work in CELADE until his retirement from the United Nations in February 1996.

Dirk Jaspers-Faijer, Coordinator of the CELADE Training and Special Programmes Area, took over direction of the Population and Technology Area and supervised the completion of winR+ and related software. He was instrumental in adding and designing many additional Assist and other user facilities and devoted many hours to testing the system.

Sebastian Carrasco, working in CELADE, intensively tested all aspects of winR+ on various sets of data and was very able in ferreting out “bugs” of varied sizes and species.

A project on Indicators of Chilean Youth for the Instituto Nacional de Juventud (INJ; National Youth Institute), which was carried out with the University of Chile in CELADE, provided an opportunity for testing and improving the new winR+ capability to work with external data files. Additionally, this project directed by Laura Ortiz, resulted in the creation of INJUm as an application of ZonPlan and winR+ to display the indicators concerning youth at the comuna (municipality) level of Chile; this application was mainly developed by Alejandra Silva with assistance from Serge Poulard.
Of the various persons who received Beta copies of the system, Edwin Saint Catherine of St. Lucia, Brian Boxhill of St. Lucia and working in the Cayman Islands, Harry Hernandez of Costa Rica, and Arie Hoekman of Holland working in Honduras, provided especially extensive and useful suggestions for improvement and error correction. Special thanks are due to Jean Claude Nadeau of the University of Montreal who helped Serge efficiently through e-mail to solve some especially complex problems in the programming of winR+.

This manual was written by Arthur Conning in English and then translated into Spanish by Claudio Meza, except for a chapter translated by Alejandra Silva. While writing the English version, much assistance was received from winR+’s creator, Serge Poulard, and from Sebastian Carrasco, Alejandra Silva, Claudio Meza and, at a further distance, from Ari Silva (who wrote part of the Redatam-Plus manual) and from Brent Hall.

The secretary of the Information and Technology Area, Adelaida Rahmer, and Alejandra Reyes, who assists her and manages the shipment of the Redatam software, provided efficient infrastructure and made the work atmosphere congenial during the long hours of the long period of incubation of the winR+ and related software.

The support of the Director of CELADE and now Deputy Executive Secretary of ECLAC, Reynaldo Bajraj, has always been unwavering on the previous and present Redatam projects and has been instrumental in making the system know widely at political as well as substantive levels throughout the region. Zbigniew Mikolajuk, Senior Programme Officer at IDRC, has been the Project’s Program Officer throughout the project and was instrumental in making the Redatam-Plus software known in Africa. Thanks must also go to Robert Valentin of IDRC, who supported all three IDRC Redatam projects that financed much of the Redatam system development, with a clear vision of the long-term purposes to which winR+ and its predecessors are directed.
How to obtain more information

The CELADE page of the World Wide Web on the Internet contains information on winR+ and other Redatam-related software or associated with it, on how to order software and on how registered users can obtain software updates. Look for Redatam-related materials in Spanish or English at the following equivalent addresses:

www.ECLAC.ORG/CELADE-Esp
www.CEPAL.ORG/CELADE-Esp

You can also communicate with CELADE at the following address:

Latin American Demographic Centre (CELADE)
Casilla 91, Santiago, Chile
Tel: (562) 210-2002 Fax: (562) 208-0252
E-mail: djaspers@ECLAC.CL

As noted in the Acknowledgments, winR+ was developed within the joint R+GIS Project of CELADE and the University of Waterloo, Ontario, Canada, funded by the International Development Research Centre (IDRC) of Canada. Descriptions of the R+GIS spatial decision support tools created by the University of Waterloo, namely, AccessPlan for health care clinic accessibility location and resource allocation, EduPlan for school location planning and management, and TourPlan for site selection and multi-criteria evaluation of tourism development, can be found along with ordering information at:

www.fes.uwaterloo.ca/Tools/index.html

You can also communicate with Professor Brent Hall, who was the Principal Investigator of the R+GIS Project in Waterloo, via:

Dr. Brent Hall, Professor and Associate Dean, Computing
Faculty of Environmental Studies
School of Urban and Regional Planning
University of Waterloo
Waterloo, Ontario, CANADA N2L 3G1
Tel: (519) 885-1211 x2399 Fax: (519) 725-2827
E-mail: gborahall@kupe.uwaterloo.ca
Chapter 1. What is winR+?

WHAT IS winR+?

Brief Summary of winR+

winR+ is the abbreviation for Redatam-Plus for Windows, where “Redatam” stands for the REtrieval of DATa for small Areas by Microcomputer. The software can be used with either Microsoft Windows 3.x or Windows 95 on all IBM-compatible microcomputers.

Compared with R+
The Redatam software first appeared in 1987 for microcomputers using DOS. The more generalized Redatam-Plus (abbreviated “R+”), also for DOS, was released in 1991. The present third generation of the software, winR+, has been entirely rewritten and further generalized to make it much more powerful than its predecessors while simplifying its use. A comparison of the features of Redatam-Plus and winR+ are given in Appendix A.

The winR+ software stores compressed data of very large datasets, such as entire population censuses with many millions of persons, households and houses. After the user defines a geographical area in terms of any set of areas down to city blocks or smaller, new variables may be derived from the database and tabulations and other outputs can be rapidly processed, all via graphical windows and without the assistance of a programmer.

While a particular user may want to process only selected sub-areas, the entire database, compressed to around 25 percent of its original size, remains available to all users. The hierarchical geographical structure of the database (see an example in Figure 2) makes it easy to produce outputs involving different levels of geography including houses and persons.
Basics of winR+

A winR+ database normally contains microdata, i.e., data (variables) that refer to individual persons, households, houses, or other elements, since this permits the production of any crosstabulations for any user-defined geographical areas. The information in a database may be multi-sectorial, in that it may come from one or more different sources like censuses, surveys and databases of administrative or other statistics.

The software can also process a winR+ database in association with an external database in one of the common formats such as dBASE, thereby avoiding the need to import small or frequently changed datasets into a winR+ database. Furthermore, if the external data are frequently used, the information on the variables of one or more dBASE files can be included in a user’s winR+ dictionary.

The winR+ software has map display facilities and can link to various Geographical Information Systems (GIS). The link allows the user to display and carry out spatial analysis of the aggregate statistics that can be produced with winR+ at any geographical level of interest and for any area(s) of interest.

Major Features of winR+

Workspace for database and user information

Each user of a winR+ database can create his/her own Workspace(s). A Workspace contains a personal copy of the dictionary describing the variables of a given winR+ dataset, the database structure with all the names and identifications of the geographical areas in the entire database, and the user’s own “documents”. These user documents include 1) the user’s selection sets for identifying geographical areas to be processed, 2) command sets for controlling the production of outputs according to the users’s specifications, 3) Geo-referenced sets — called AreaLists — for transferring data from winR+ to a GIS or for carrying out processes with an external file and 4) map definitions (when an optional module is available).

The Workspace allows individual users to define new variables, etc., without having to take precautions to avoid conflicts with other persons using the same database. Alternatively, a user may wish to create a separate Workspace for each of his/her projects using the same database. Note that the actual data files of a database are not in the Workspace so
Chapter 1. What is winR+?

that various users can all share the same often very large winR+ databases.

The database structure and the dictionary of the variables at each geographical level can be displayed in separate graphical windows at any time so that the information is always available if required.

Geographical selection

All processes done in winR+ require that the user first define the ad hoc geographical area for which the tabulations or other outputs will be produced; this is known as the user's Selection Set (SelSet) or selection area. In this way, although winR+ may have data for an entire city or country, it knows for what specific area(s) the user wants outputs. Figure 1 gives an example where a sub-set of blocks in a city have been selected for a micro-planning project. winR+ provides graphical aids to identify and select, for example, the blocks, making up the planning area. It is also possible to use a Command Set written in the winR+ language to define a Selection Set based on calculated criteria, such as all blocks with more than a given percentage of the children living without a father in the house (see the sub-section on the command language below). A module also allows selection by pointing to areas on a map.
Figure 1. A planning area around a Metro station. Although the winR+ database may have data for the entire city—or country—, if the indicated portion is taken as a Selection Area, winR+ will jump directly to and process only the data of this area, thereby producing outputs very rapidly. (mapSelS4.wpg).

A winR+ database can also be structured hierarchically by non-geographical criteria. However, most databases are structured entirely by one or more geographical hierarchies, in which case the "area" selected by a user refers to an expanse on a map.
The statistical processor language

The user creates a *command set* that directs winR+ to produce the outputs desired — crosstabulations, frequencies, averages and lists of areas with variables describing each of the areas, which are known as AreaLists. The winR+ software has a new language, which does essentially all that the previous Redatam and Redatam-Plus language can do and much more, and, in addition, is more intuitive to write and easier to “read” once a command set is written.

The winR+ language consists of just three basic commands, **RUNDEF**, **DEFINE**, and **TABLE**, each of which can have various sub-CLAUSES and modifiers. A separate **DEFINE** command is used for each new variable created or receded and a separate **TABLE** command is used for each output. Frequently used variables can be saved to the winR+ database. New functions and extensions of the language can be easily incorporated.

*Assist* windows can be opened to help the user create command sets at different levels of complexity. For example, there is a visual *Assist* window (*EasyTabs*) to guide the user in the rapid creation of tabulations of database variables for the user’s selection area, without entering any commands at all — these are generated automatically. There are also *Assist* windows to aid the creation of new variables and various outputs.
using the mouse to click on variable names and command keywords. The Assist to produce outputs, and particularly the EasyTabs, will be of special value to new uses and the important class of occasional user who may not recall the specifics of the command language.

**An open system: winR+ and external databases**

The user's command set is sent to the winR+ statistical processor, which is the statistical engine that actually processes the database for the user's selection set. While the request will normally involve a winR+ database, which may be multi-sectoral with data from different sources incorporated, winR+ is also an open system. This means that it can at the same time access a winR+ database and/or information from external non-winR+ databases, which may have one of a number of formats that winR+ supports, most importantly, dBASE (.dbf) and Microsoft Access (.mdb).

Thus, taking advantage of the openness of winR+, a planner might derive a socio-environmental indicator describing city blocks for a municipal disaster preparedness plan by processing together 1) a population and housing census winR+ database to create an index of poverty for each block of the municipality based on data at the household and person levels, and 2) an external dBASE database maintained by the municipality with information on the environmental hazard levels estimated for each block. The latter need not be physically included in the winR+ database, thereby allowing winR+ to use data from periodically updated sources without having to modify the winR+ database, itself, which was designed for normally unchanging data such as that of a census.

**Hierarchical processing**

The hierarchically-structured winR+ database allows the derivation of results based on two or more levels (called "entities" in winR+, see page 22) more easily than most other tabulation software. Thus, the creation of a table to see whether women or men (person level) are more likely to live in houses without running water (house level) simply involves asking for a TABLE of sex BY water.

As evident from some of the examples already given, winR+ facilitates the creation of statistics for aggregations of individuals by taking advantage of the hierarchical organization of its database by geography (see Figure 2; note that a database may be much more complex with various branches, such as shown in the Figure on page 25). Thus, an
aggregate statistic like the percentage of children under 5 years of age in
each district who live in houses without potable water may be easily
determined from information on individual houses and persons and these
results can be sent to a GIS for cartographic display and spatial analysis.

**Outputs: Frequencies, Cross-tabulations, Averages and AreaLists**

The outputs created by the `TABLES` command — crosstabulations, fre-
quencies, averages and lists of areas with variables describing each of the
areas (AreaLists) — may be displayed on the monitor, printed, or sent to
a file in a variety of formats. AreaLists, in which each row refers to a
different area, say, the blocks of an electoral district with variables de-
scribing them in columns, can be produced in different forms, among
others, as a file that becomes part of a GIS attribute table in a GIS using
common database formats like dBASE or Microsoft Access. When
election of the sub-areas making up a selection area requires the prior
derivation of new variables, the user can direct the `winR+` processor to
produce the required Selection Set.

**Speed of processing**

The efficiency with which records are processed is very high, although,
of course, the actual speed will depend upon the microcomputer CPU, the
hard disk access and other parameters as well as upon the specific
commands to be processed. On a Pentium 100 microcomputer with
Windows 3.x, a 2-way crosstabulation of 201,000 records took 11
seconds.

*Compared with R+*

The speed is considerably faster than R+. For the same task and microcomputer as
mentioned in the previous paragraph, the time was 43 seconds. In general, using Windows
3.x or 95, the speed of processing with `winR+` is around 3 to 4 times faster than when
using R+.

**The winR+ Data Link: GIS and more**

The processing of a `winR+` database together with an external database
requires that a link be established between the geographical codes of the
`winR+` database and those of the external database or a GIS. Assisting the
user to do this transparently and correctly is one of the major functions of
the **winR+ Data Link**.

The **winR+ Data Link** works closely with the AreaList output of the
statistical processor. Since the rows of an AreaList each refer to an area at
a given level of geography—e.g., each row might have information on a
different municipality within a country—, the value of a given variable or
attribute for each area can be displayed on a digital map, allowing the
user to ascertain physically adjacent groupings of provinces and other
spatial patterns. The Data Link moves the AreaList to a GIS (or an
external database of another software system), puts the output data from
winR+ into the format required for the external system, and utilizes an
appropriate key to match the winR+ geographical codes for the areas with
the corresponding ones in the external file. It also checks for errors, such
as those that occur when some winR+ database areas do not have
counterparts in the external file (e.g., a map) or vice versa.

Prior to moving the AreaList to the GIS or other external system, the
columns of the AreaList, each of which refers to an aggregate variable or
field with values for each area, can be manipulated in a “spreadsheet”
within winR+ to produce new variables.

Multilingual facilities

The winR+ software has Spanish and English user interfaces, which can
be changed from one to the other at any time. Furthermore, the software
has been designed to facilitate easy and systematic translation into other
languages. Note, however, that the statistical processor command lan-
guage, which uses English keywords, is the same in all languages. Later
versions of winR+ may provide commands in the language of the user
interface.

Data security

The data have three types of protection. First, during the creation of a
winR+ database, the database administrator can declare as “unidentifi-
able” or “non-selectable”, say, the house, household and person levels,
while making the block and higher levels “identifiable” or “selectable”,
that is, the user can identify specific elements at these levels by name
and/or their winR+ codes. Second, there is a system of passwords that the
database administrator or a user can set to make a Workspace and
associated dictionary unavailable to those without the password. Finally,
to avoid users making changes in the database or its dictionary, the
administrator can set a password that allows only he/she to make
changes; of course, the user can make changes in his/her own Workspace,
but these do not affect the database or its dictionary employed by other
users.
Chapter 1. What is winR+?

The winR+GIS decision support system tools

The winR+ software is a generalized software system that can be utilized for a large variety of purposes and with datasets from many different fields. On the other hand, many workers in specific fields, such as educational or health facility planning may not want to learn generalized software, but still may need winR+ capabilities coupled with a GIS for local planning in a package oriented to their specific problems. To this end, a joint project of CELADE and the University of Waterloo, Canada, funded by the International Development Research Centre (IDRC) — see the Acknowledgments for the full list of donors which have supported this and the Redatam efforts in general— developed winR+GIS decision support tools in a number of fields.

In particular, the University of Waterloo created AccessPlan for family planning and primary health care planning, EduPlan for educational planning, and TourPlan for tourism development planning with an environmental focus; while CELADE developed ZonPlan to assist planners to create indicators for identifying target populations. Tools for other fields or further adaptations of these can be developed using the winR+ building blocks (see next subsection).

The winR+GIS tools are not included when the full winR+ software package is ordered, but may be obtained separately from CELADE or the University of Waterloo.

The winR+ building blocks

The winR+ software, along with the winR+GIS tools mentioned in the previous subsection, have been designed in terms of reusable components, each of which can be employed in other software. Thus, there are building blocks for making new winR+GIS tools or other winR+ based applications. Since a reasonable level of programming knowledge is required to use the building blocks, the present manual, which is directed toward data users, is not concerned with this subject.
Where Does the Database Come From?

One of the major objectives behind the creation of the Redatam software was, and still is among others, to facilitate the utilization of the housing and population census data in each country at the local level. As the creation of a census database is an undertaking that is normally done by the National Statistical Office of the respective country and requires more programming skills and hardware than are likely to be available to most winR+ users, they will normally obtain all or a portion of the national census from their National Statistical Office.

The winR+ software has been designed to use any existing Redatam-Plus database —or Redatam 3.x database after conversion to Redatam-Plus—. A few mouse clicks will bring the existing Redatam-Plus dictionary into your Workspace and make the necessary conversion. The data files, themselves, can be used directly by winR+ without conversion.

The first release of winR+ does not include a module for creating new winR+ databases. For now, new databases and their associated dictionaries can be created with Redatam-Plus (DOS) and then brought into the winR+ Workspace. An early update of the first release will include database creation facilities.

So that you can experiment with the winR+ software before obtaining or making your own database, the software comes with a demonstration database, New Miranda, upon which the tutorial and examples throughout this manual are based.

Who are the Intended Users of winR+?

Three general classes of users are expected to work with winR+, namely:

**End-users**

Most users of the system will be end-users, that is, persons who are concerned with the exploitation of data in winR+ databases and associated external data files. Such persons, are not expected to have any computer science, data administration or programming knowledge, and may or may not have worked with the previous generation, Redatam-Plus.
Chapter 1. What is winR+? — 11

As noted earlier, many winR+ databases are very large and their creation requires programming knowledge and experience. Hence, end-users will normally have the winR+ databases that they require installed in their institution or will obtain an existing database already created from someone else.

This manual, Basics of winR+, is oriented to End-Users, although programmers and database administrators with extensive experience with Redatam-Plus will find it useful for learning about the many new features, particularly the new command language.

Database administrators

The second category of users is much smaller, consisting of persons who make and administer winR+ databases. They normally will need some programming knowledge and experience in creating, checking and maintaining databases. This experience is important since winR+ databases, like most other databases, are employed by end-users who are likely to take for granted that a given database is logically consistent and free of non-substantive errors.

The assistance of a database administrator or programmer may be needed in other database related aspects. In particular, some end-users may not be able to create and verify the link between a winR+ database and a GIS attribute file so that outputs can be displayed and analyzed in a GIS, and some end-users may also be uncomfortable setting up a link to jointly process a winR+ database and external files.

Builders of winR+GIS tools and other applications

The creation/adaptation of winR+GIS tools such as those described on pages 9, using the building block components outlined on page 9 will be done by persons with a moderate to high-level of programming knowledge, depending on the problem concerned. This will normally require using a programming language like Microsoft Visual Basic to connect together the existing components or to create new ones. A separate manual will be issued for this purpose.
Chapter 2. Installing \textit{winR+}

INSTALLING \textit{winR+}

Requirements for using \textit{winR+}

User knowledge

This manual assumes that you are familiar with Windows 3.x or 95, depending on which is used in your computer. Aspects of Windows are only treated when they have a special relevance to the \textit{winR+} software.

System requirements

Recommended configuration

When \textit{winR+} is used to process a "small-area" from a larger database, the computer need not be very powerful (see the minimum configuration shown in this section). However, there are times when you may wish to process an entire city, create indicators for each block within a municipality, or use the output with a GIS, for which purposes a more powerful computer is convenient. The following is a reasonable hardware and software configuration for such use:

- A microcomputer or notebook with a 133MHz Pentium-based system CPU or higher.
- Windows 95; or Windows 3.x, running 386 Enhanced Mode, with DOS 6.0 or higher.
- 16 MB RAM or higher.
Basics of winR+

- A mouse or pointing device.
- Color monitor, Windows compatible.
- A hard disk with at least 10MB of free space for the software and 1.5MB for the demonstration database. The total amount of space required will also depend, of course, on the size of your database(s) —as a rule of thumb for most census databases, winR+ requires around 20MB for each million persons, which takes into account their housing and household information—.
- 3.5 inch high-density floppy drive to read in the software, make back-ups of user programs, etc.
- (Optional) CD-ROM to download databases or software (winR+ may eventually be available in this format).
- (Optional) Printer. Black and white will be used for tables and other textual output, but if purchasing a printer, it is worth considering color, since these are now inexpensive and you may eventually wish to produce color-coded maps based on winR+ outputs or to use winR+GIS tools, which have GIS capabilities incorporated.

Minimum configuration

- An IBM fully compatible microcomputer with a 486 CPU with a math coprocessor (a 386 CPU with a math coprocessor will operate, but is likely to be slow for most work).
- Windows 3.x, running 386 Enhanced Mode, with DOS 5.0 or higher.
- 8 MB RAM.
- A mouse or pointing device.
- Monitor, Windows compatible.
- A hard disk with at least 10MB of free space for the software and 1.5MB for the demonstration database. The amount of space required will also depend, of course, on the actual database used —see the rule of thumb suggested in the previous configuration—.
- 3.5 inch high-density floppy drive to read in the software, make back-ups of user programs, etc.
Chapter 2. Installing winR+ – 15

Installing winR+

Installing winR+ in Windows 3.x

Before installing winR+, please look through the Readme file that is included on the first diskette of the installation diskettes. It may indicate variations in the setup procedure outlined below:

1. It is recommended that you exit all programs that are running on your system, except Program Manager.
2. Insert Disk 1 into your floppy disk drive, or if you have winR+ on a CD-ROM, insert it into the CD drive.
3. Choose File from the Program Manager main menu and Run from the next menu that pops up (abbreviated in this Manual as File|Run). The Run dialog will appear.
4. Type a:setup for a floppy, or d:setup for a CD-ROM (substitute the correct drive letter for your input device if not a: or d:).
5. Follow the instructions on the screen to complete the installation. For convenience in using this Manual, it is recommended that the directories suggested during the installation be accepted, although the use of different directory names has no effect on the operation of winR+.

Note that SHARE must be included in your autoexec.bat file. Unless you have problems, you need not be concerned, since SHARE is usually included with microcomputers running Windows 3.x. If it is not included you should add the following line in your autoexec.bat file:

SHARE /i:500 /f:5100 [note that i:500 has a lower-case L]

Installing winR+ in Windows 95

Before installing winR+, please look through the Readme file that is included on the first diskette of the installation diskettes. It may indicate variations in the setup procedure outlined below:

1. It is recommended that you exit all programs that are running on your system.
2. Insert Disk 1 into your floppy disk drive, or if you have winR+ on a CD-ROM, insert it into the CD drive.
3. Click the Start button, then click Run from the pop-up menu.
4. Type a:setup for a floppy, or d:setup for a CD-ROM (substitute the correct drive letter for your input device if not a: or d:).
5. Follow the instructions on the screen to complete the installation. For convenience in using this Manual, it is recommended that the directories suggested during the installation be accepted, although the use of different directory names has no effect on the operation of winR+.

Launching winR+

The winR+ system is represented as an icon within a Program Manager Group in Microsoft Windows 3.x, and in Windows 95 usually as a shortcut with the same icon. Since the suite of R+GIS Decision Support Tools, including EduPlan, AccessPlan, TourPlan and ZonPlan are meant to be used with winR+, the winR+ icon on some computers may be found with the R+GIS Tools (see page 9).

If the Program Group is not already open on your Windows desktop, double click the icon to open it. Double-click the winR+ icon to launch winR+.

Exit winR+ as any other Windows program, e.g., click File|Exit on the main menu. It is desirable to Exit before shutting down Windows.

Customization

[Section Optional on first reading]

You may wish to skip this section on first reading, if you have not used Redatam-Plus in the past or are relatively inexperienced with statistical processing software. For these cases, the standard defaults are available; you can customize for your own needs and preferences at a later date.

At any time, you can make changes in a number of aspects of winR+ that do not affect the substantive results of processing data, but do affect your interaction with the system. You can enter your own preferences by clicking on Tools|Preferences on the main menu. This opens up the Systems Options window, which as seen in Figure 3 has a number of Tabs at the top. These and their contents are straight-forward and will only be briefly treated here:
a) General Tab:

- **Default working directory:** This is the directory (folder) where a file that you create with winR+ is placed if you do not explicitly state a different directory. Initially, the working directory is set within Windows 3.x or 95, when you install winR+. You can change the working directory here by typing in a path to another existing directory or use the Locate button to select another existing directory.

- **Do not show splash screen:** This is the initial screen with the winR+ logo that is presented on startup. You can start work more rapidly by checking this box; if you want to see the information on this screen at any time, you can click Help|About on the main menu.

- **Load last Workspace on starting winR+:** You may find it convenient to check this box, if you are working for various days with the same Workspace (where you keep your copy of the dictionary, your Command Files, Selection Sets, certain outputs, etc.). If this box is not checked, you have to manually select and load the relevant Workspace each time you begin a new session with winR+.

- **Password:** This allows you to put a password on access to your Workspace. In effect, if a person is locked out of the Workspace referring to a given winR+ database on a computer, the person cannot use the database. To create a password, click on this button and follow the simple instructions and then Save and Exit. To remove a password, you must enter with the existing password, leave blank the spaces for the new password and its repetition, and click Save and Exit. Of course, it is your responsibility to remember your password.
b) Printing and Editing Tabs: These alter the fonts used on 1) all editing screens; and 2) all printouts of results, command sets, etc. Figure 4 shows the screens involved for changing the font for Printing (the Editing screens are identical). The Printing and Editing fonts can be different, but it is usually best to keep them the same.

![Figure 4](siPrefPr.wpg)

Figure 4. Window showing the Printing Tab and the window that opens to set the font for printing; the Editing Tab is similar and normally should be set to the same font and size as for printing.

c) Language: (not shown) English and Spanish will be available on early versions of winR+, and later, probably also French and maybe
other languages. The buttons on this screen change the software from one language to another.

*The Workspace must be closed before accessing the language change tab.*

d) **Work Mode:** (not shown) This cannot be changed by a user. Only a systems administrator who has the necessary password can change the mode to allow changes in the text of one or more of the languages, etc.

*[End of section which is optional on first reading].*

---

**The New Miranda demonstration dataset**

The winR+ software comes with a full demonstration dataset, consisting of a winR+ database with multiple branches and some external data files relating to the same geography, all for the hypothetical country called New Miranda. As the Tutorials (see Chapters 4, 5) and most of the examples in this Manual are based on this demonstration dataset, it is recommended that it be installed initially with the software, since it can be easily erased after you become familiar with winR+.

*Compared with R+*

The New Miranda dataset is based upon the same database included with Redatam-Plus. To distinguish the two, the winR+ demonstration database is identified as WMIR, while the older database was identified by NMIR. The winR+ demonstration dataset is somewhat smaller than the older one, since some of the many multi-sectorial branches were eliminated. Also some of the branches are now separate, non-Redatam, dBASE and other files so that you can practice working with external files, one of the major new capabilities of winR+. The data in the wmir database have been "updated" and now are based upon the hypothetical national censuses of 1990 and 1980.

[wr1ic20.man c:\winplus\wrman1]
Chapter 3.  

BASIC winR+ CONCEPTS

It is convenient for you to have an initial set of concepts and definitions to facilitate conversation during the “walk-through” and tutorials of the next chapter. Additional concepts will be defined when required in this Manual.

**winR+ database concepts**

The basic concepts of a winR+ database can be explained in terms of a simple census-like database as represented graphically in Figure 2 on page 4 and shown in more detail in Figure 5 on page 22.

**Entities and Elements**

Each of the boxes on the left in Figure 5, such as *Province* or *District*, represents a geographical level that is hierarchical to the previous level,
e.g., districts are smaller areas within provinces. A specific set of
districts, each with its own name and unique code in the database, makes
up a province. Each level—the boxes in the Figure—are called Entities
and the set of areas within a given entity are called entity elements. On
the right side of the Figure, you will see that Santiago and Lautaro are
two of the district elements, which comprise the Province of Valdivia.
Valdivia is one of the set of province elements making up the entire
country.

Figure 5. Entities defining a database hierarchy and some of the elements
making up each of the entities.

The database under discussion has a single branch since there is only one
path from the lowest to the highest, or root, entity, which always has a
single element, the entire country in the case shown. Of course, the root
entity of other databases might be a city or a municipality within a city.

A winR+ database, like the New Miranda wmir database that
accompanies the software, can have various branches so that two or more
censuses, surveys and other data can be included in the same database.
Figure 6, as diagramed in the winR+ Dictionary window, shows how the
New Miranda database entities are structured. Note that some branches
have sub-branches lower down, e.g., the 90block entity divides into two
branches for the 1990 census entities of 90housin and school, where the
later are the school(s), if any, on each block.

Selective and unselectable elements

You will note in Figure 5 that the individual elements of the entity,
Block, are without names, but still have numbers to distinguish one from
another, i.e., Block1, Block2, etc. In census geographical hierarchies,
there maybe various levels of entities on a branch without element names
but with codes which connect them clearly with specific areas on a map.

The last two levels, House and Person, are also entities. The houses
within a block are the house elements of the particular Block and
similarly, the persons within a given house are the person elements of that
house. However, normally, House and Person entities (and Household
entities when included) are different from the higher entities in that they
have neither names nor identifying codes in the database. Hence,
individual instances of these are unselectable in winR+, although they
can be counted during a process. Higher level entities, with named and/or
coded elements, are selectable. In effect, you can produce results for
individual elements of selectable entities, but cannot produce results for
individual unselectable entities, e.g., for a specific house in the New
Miranda database. Of course, you can produce tabulations of houses or
persons for a selected area since specific houses or persons are not
individualized.

Based on data security and substantive needs, the database creator can
define any given entity level and lower as unselectable. There is a
tradeoff between security and usefulness. For example, if the Block entity
is made unselectable, it is not possible to derive an indicator of poverty
for each block from housing and person information and then map them
to see if there are "pockets of poverty" of adjacent blocks with similar
poverty levels.
Variables

Information items about individual elements of an entity is stored in variables. Hence, each entity has its own list of variables, and each element of the entity has its own set of values of these variables. For example, the entity Person might have, among others, the variable sex and one of its elements may have the value of sex = female. Any entity can have variables, e.g., Province might have variables like average rainfall per year and the percentage of unmarried mothers in the fertile ages. A variable is identified in winR+ by the entity to which it belongs and the name of the variable, with a period between the two, that is, EntityName.VariableName

Thus, the variable sex might be identified as persons.sex in a database. (this will be explained in more detail in a Tutorial).

The values of a variables usually are divided into categories or classes; see the Glossary in Appendix C.

Some definitions for working with winR+

Dictionary

The dictionary contains all the documentation — sometimes called metadata — describing a winR+ database that you and the software each require to work with a given database. This includes information on the hierarchical structure of the database, each branch of the hierarchy, each entity, each entity element, and each variable. You can view this information by calling up the Dictionary window.

Database structure

The database structure defines the hierarchical relationships among the entities. You will need to view the structure when selecting the area(s) of interest with which you wish to work.

Selection set

winR+ is designed to facilitate working with sub-areas of an entire database — see Figure 1 on page 3. Hence, in any given process, you must always begin by telling winR+ what part of your database you wish to
work with. In the usual database defined in terms of a collection of specific geographical areas, this is done by specifying a Selection Set, which consists of the areas of interest (i.e., entity elements, like provinces, or districts or city blocks). Any identified area, including the entire country of say, a national census, or any combination of such areas, may be used to define a Selection Set, which is often abbreviated as SelSet in this Manual.

The name “Selection Set” is used rather than the more obvious and less pretentious “Selection Area,” since it is possible to structure a database such that non-geographical criteria define the SelSet. For example, a vital statistics database might be organized by year, in which case you would have a SelSet that referred to a time interval(s).

![Diagram](image.png)

Figure 6. New Miranda, a multi-branch winR+ database.

**Command set**

The instructions to tell winR+ what output(s) you want for a given selection set are known as a command set — abbreviated as CmdSet in this Manual —, which is written using the winR+ language. After the software checks (compiles) your command set to ensure that there are no syntax
errors, you can then run the process to receive the output. In some situations, winR+ will write the command set for you after you indicate what you require on a simple form.

Tables: Frequencies, Crosstabs, Averages and AreaLists

The standard outputs, called Tables in winR+, of a command set for a given selection set are Frequencies, Crosstabs, Averages and AreaLists. The first three are standard outputs of most statistical software and do not need a definition here, the fourth type is described in the next paragraph.

AreaLists

AreaLists are a special type of crosstabulation, in which the rows of a table each refer to a different geographical area, such as the districts in a province or the city blocks in a health planning area, and the columns to variables (see Figure 39 on page 103). AreaLists are especially important in winR+ since they can sent to a GIS for cartographic display and spatial analysis, used to produce new Selection Sets based upon calculated criteria, manipulated in other software, such as spreadsheets, or written, for example, as dBASE files for use in commercial database programs.

Run

A Run refers to all that winR+ does to produce the results for a single Command Set. A Command Set can involve the creation of many derived variables and the production of various output tables as defined above. However, only one Selection Set can be processed in a given Run. Thus, if you wish to carry out the same set of tabulations on two planning areas, you normally must make two runs, each with its own Selection Set. A “Run” may also be called a “job”. Prior to running a Command Set, winR+ compiles the instructions to check that the syntax is correct; if not, it indicates where there are errors.

Workspace

The Workspace has the information —sometimes called documents—that you require to exploit a database, that is, the dictionary, the command sets and the selection sets. In addition, you may also wish to keep your AreaList results in your Workspace, since they may be required for
sending to a GIS or other external system or for displaying on maps whose definitions are also kept in the Workspace. Since the Workspace helps to keep together all the relevant materials for working with a given database, you may find it convenient to create a different Workspace for each project that you have, even if the database is the same — only the dictionary, which takes up relatively little space, will be duplicated, since the datafiles, themselves, are not included in the Workspace.

The Workspace is a single .mdb file which holds the dictionary, command and selection sets, etc. This makes it a simple package for transporting a Workspace from one computer to another or for backing up periodically.

**Assist**

*Assist* windows, often called Wizards in other Windows software, are available at many points in the *winR+* software to help you create command sets or to carry out procedures to produce specific results.
Chapter 4. A walk through \textit{winR+}: [Tutorials]

A WALK THRU \textit{winR+}: BECOMING FAMILIAR
[Tutorials 1-2]

The walk through \textit{winR+} in this and the following chapter has two purposes. First, it provides Tutorials for teaching new users how to obtain commonly required results from the software, and second, it is a vehicle for showing potential users what \textit{winR+} does and how it is done. The Tutorials of Chapter 4 will show you how to navigate in \textit{winR+} and carry out rapid tabulations of selected geographical areas without having to know the \textit{winR+} language. The Tutorials of Chapter 5 will introduce you to the \textit{winR+} language—which is different from that of Redatam-Plus—and show you how to obtain the various \textit{winR+} outputs.
As noted previously (see page 19), the Tutorials are based on data from the fictitious country called New Miranda. The Tutorial sections marked with [Optional] can be ignored on the first reading.

Compared with R+

To simplify the database and reduce its size, various changes have been made in the winR+ New Miranda database when it was “updated” from that was provided with the Redatam-Plus DOS software. Some branches in the rather overly complex original version have been eliminated and some branches have been converted into separate .dbf files to provide examples, in a later chapter, of how to work with the new feature for processing external files that are not physically integrated within a winR+ database.

You should be familiar with the basic concepts and definitions used in winR+ before starting the tutorials. If you have not already done so, it is strongly suggested that you read Chapter 3 which outlines necessary concepts for working efficiently with winR+.

The winR+ software, like many other Windows-based systems, often allows you to carry out the same operations in more than one way. To keep this basic manual a reasonable size, it often outlines one approach to a given activity in detail and only mentions, if so, other alternatives.

TUTORIAL-1: Navigating in winR+

In this first Tutorial you will become familiar with the look and feel of winR+. In particular, you will learn how to:

- Identify the components of the Main screen of winR+ and the icon bar.
- Open and use an existing Workspace.
- Call up dictionary and database structure information.
- Use the Help facilities of winR+.

The winR+ main screen

After you have started winR+ (see page 16) and opened a Workspace (refer to the next section), you will see the winR+ working environment or main screen on which you will open and use various smaller windows to carry out the actions you wish on your data. The main screen will look
like that shown in Figure 7. Note that you can obtain a short description of any icon by resting the cursor on it.

As other Windows-based software, winR+ has a Title Bar, Main Menu Bar, buttons, combo boxes and other components of the Graphical User Interface (GUI) that are normally operated by a single click on the relevant object with the left mouse button. This is taken as standard in this Manual. Only when a double click or a right mouse button click is required will it be explicitly stated.
The Icon (Tool) Bar and its icons

New Workspace document
Open Workspace document
Print
Show Command Sets in the Workspace
Show Selection Sets in the Workspace
Show AREALIST's
Display Map
Compile the Command Set in the active Command Editor
Run the Command Set in the active Command Editor
Abort the Run of the Command Set
Open the Dictionary, including database structure
R+GIS Link
Help

The icons seen on the Icon Bar will vary with the windows that are open.
Living in a Workspace

All your work using winR+ for a particular database is done in a named Workspace that keeps track of your documents and holds the data dictionary and other required information. The Workspace window shown in Figure 9 contains the Command Sets, Selection Sets and other "documents" that you create for a given winR+ database and provides the means to access them. Different databases must have different Workspaces. However, you can also create more than one Workspace for a given database and it is recommended that you create a separate Workspace for each different project that uses the same database.

The Tutorials in this and the next chapter are designed to be used with the New Miranda demonstration database and related Workspace that come with the software. On the other hand, creating a new Workspace, converting a Redatam-Plus database dictionary into a winR+ Workspace, and related manipulations are described in a later chapter (see page 138).

![Workspace Window](siWSw.wpg)

Figure 9. The Workspace Window. The open combo box shows the document types available; as Selection Sets has been chosen, the window lists the SelSets that are in the Workspace.

The Workspace window

After starting winR+, the main screen will appear—if the logo screen appears, click the OK button—. In this Tutorial, if it is not already on your screen, you should open the New Miranda Workspace that is in your work directory after installing winR+. This is done by clicking the Open
icon. Alternatively, you can do this by clicking on File on the
Main Menu and then clicking on Open in the pop-up menu that appears
(this sequences is abbreviated: File|Open). Select the wmiri.mdb
Workspace and after a few seconds the Workspace window will appear
(see Figure 9). wmiri.mdb is the English-language New Miranda
Workspace and wmirre.mdb, the Spanish-language one; the content is the
same in both.

A list of recently-used Workspaces is given at the bottom of the pop-up
window when the File menu has been clicked; clicking on one of these
can also be used to open an existing Workspace. If you plan to use the
same Workspace for various sessions, you will find it convenient to
customize winR+ to start-up with the Workspace that you last used in the
previous session (click Tools|Preferences on the Main Menu and then in
the General Tab window, put a check in the box for Load last
Workspace on starting winR+. Click the Save and Exit button).

The winR+ installation diskettes may also have English and Spanish
versions of the Workspace with all the CmdSets, SelSets, etc., described in
the Manual. Please read the Readme file on the first installation diskette
for information on the presentation of the New Miranda database and
Workspaces.

Occasionally, you may find that the Title Bar of the Workspace, which is
dragged to move the window, has disappeared under the Main Menu
(Tool) bar, making it impossible to move the Workspace window. When
this happens, a scroll bar will appear on the right side of the screen.
Clicking on the top arrow of the scroll bar, will bring down the full
window.

Workspace documents: The Workspace of the present release of
winR+ holds a total of four types of “documents”, namely Selection Sets,
Command Sets, AREALISTs and Map Definitions. The combo box
Command Sets, which shows the types of documents after the
down arrow is clicked (see Figure 9), can be used to select the document
type that you wish to work with. The icons on the icon bar, ,
, can also be used to display the Command sets, Selection
sets and AREALISTs, respectively. When initially created, the wmiri
Workspace contains only the Selection Set (SelSet) *All* for the entire country.

**To view a document:** To view, for example, Selection Sets. In the combo box in the Workspace set the Document Type to *Selection Set*. To view a particular SelSet, first highlight the SelSet of interest and then click on the **Open** button. To highlight the SelSet or any other Workspace document, move the cursor to the left column of the row until an arrowhead appears and then left click the mouse. The row will change its color indicating that it has been selected. Holding the **CONTROL** (CTRL) key down when performing this action allows different non-adjacent documents (rows) to be selected; if the set of documents that you want are contiguous, click the left column of the first and then hold down **SHIFT** and click the last of interest.

**Importing and exporting documents:** Since the documents are all stored in the same Microsoft Access database for a given Workspace, there are import and export facilities for exchanging documents among Workspaces and for producing ASCII text files which can be edited externally and sent as files to other users. The procedures will be briefly outlined in a later chapter (see page 142).

*Note that the Workspace window is “sticky”. That is, any changes that you make in its size or position are remembered in future sessions. These changes are made, as usual in Windows software, by placing the cursor on the edge of the window until a double arrow appears and then dragging the edge in or out, or diagonally if the corner is used. The column size can also be adjusted by doing the same on the column separator lines between the column names. Similarly, the row height can be changed by placing the cursor on any row in the right-most column and dragging the double headed arrow to make the rows wider or narrower.*

**To gain space on your screen** when you do not need the Workspace window visible, you may wish to minimize it by clicking in the upper right corner of the Workspace window. The minimized Workspace will appear at the bottom right of your screen as .

Double click on this to restore the Workspace to its previous size and position.
It is strongly recommended that you BACKUP your Workspace periodically and systematically since it has all your Command Sets, Selection Sets and various data sets. At the minimum, backup frequently to another directory on your hard disk since this will allow you to recover accidentally erased CmdSets, etc., or possible corruptions of your Workspace(s). Of course, much better, you should backup frequently to another medium such as a floppy disk since that will also protect you from disk crashes.

**MAKING A WORKSPACE BACKUP:** With winR+ closed, go to whatever system you use in Windows to manage files — e.g., Windows Explorer in Windows 95, File Manager in Windows 3.x, etc. — and copy your Workspace file to a floppy disk. The file has the name of the Workspace and the extension .mdb and normally resides in your winR+ work directory.

**Viewing the dictionary and database structure**

The database dictionary, which is incorporated within your Workspace, may be accessed by clicking on the **Database** | **Dictionary** menu or on the Dictionary icon ![Dictionary Icon](image). The right side of the dictionary window that opens — see Figure 10 —, 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 box on the left-side contains the list of variables describing the currently highlighted entity on the right side. Click on another entity name to change the current entity. The list of variables in the left box is then changed accordingly. Double click on an entity to open a window displaying information about the entity. Similarly, double click on a variable name such as **90attend** (on the left-side list) to open a window with two tabs displaying the information about the variable (shown in Figure 10).

The Variable window has three tabs. Clicking on the first provides **General information** (Figure 10) of immediate interest to end-users since it gives the categories of the variable with their codes and the range. The **Documentation** Tab window may have narrative information concerning the variable and the **Storage/Variable** window provides technical information.
Chapter 4. A walk through winR+: [Tutorials] — 37

The information on the first Tab of the Dictionary includes the range of the variable, i.e., the minimum and maximum values that are legitimate codes; the values of the Missing and NotApplicable codes are also given. The latter two should, in principle, always be outside the range to allow winR+ to treat them correctly in output tables. If this is not the case due to an error in the database construction, when you access the Dictionary, you will receive a warning message like the following from winR+

so that

you use such a variable with care. Careless use, for example, of a children-ever-born variable with 99 as NotApplicable, but within the Range, will give you very high fertility

Branches of the dictionary tree may be collapsed or expanded by clicking on the left side of the name of an entity. If you collapse an entity with many sub-branches, you either have to reopen each separately or you can bring back the fully expanded structure by closing and then reopening the Dictionary window.

Obtaining Help in winR+

The winR+ system provides assistance to users through Help on the Main Menu and the Help icon . The Main Menu Help has the following entries for several different types of assistance that range from very general to highly specific, namely:

<table>
<thead>
<tr>
<th>Contents</th>
<th>This provides information on how to carry out the major functions of winR+.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>This allows you to Search for Help on... a topic of interest; is similar to, but more convenient than an index in a published manual; You enter a word or phrase and the system gives you all the entries that are similar. After selecting one, you can bring up a Help screen with the information.</td>
</tr>
<tr>
<td>winR+ latest</td>
<td>This provides information on additions, changes and known “bug” (problems) in the latest version or update of winR+. This manual is based on Vers. 1.1.</td>
</tr>
</tbody>
</table>
About

A single screen gives the Version number and Revision number of the winR+ software that you are using, the logo, and information on the project within which winR+ was developed, the institutions involved and the donors who supported the work, as well as the address of CELADE. The latest E-Mail and Internet Web addresses are also given.

In many situations, you can obtain context-specific Help by pressing the F1 key; Help on error messages is often available through this procedure.

Figure 10. Dictionary window (on the left) showing the tree structure of the database entities. The variables in the left column are those of the highlighted entity 90person. A variable 90attend of this entity, in turn, was then highlighted and double clicked to open the right window with the descriptive name, categories, etc. of the variable.

{siDicVar.wpg}
TUTORIAL-2: Simple information retrieval

Using your Workspace, you can retrieve information from a database by telling winR+:

1. What areas to process: Creation of a Selection Set.
2. How the information should be processed: Creation of a command set or its equivalent.
3. When to carry out the execution of the process.
4. What results to display.

In this Tutorial you will learn how to use winR+ facilities to obtain information from a winR+ database. In particular, you will:

- Use the Selection Editor to create a Selection Set comprising the part of New Miranda for which you need to obtain information;
- Obtain simple frequencies, cross tabulations, averages and counts without having to write any commands (use of EasyTabs);
- View your results.

Creating a Selection Set

The winR+ software was designed for quick access to small-area data contained in databases comprised of very large areas like whole countries. Therefore, before requesting an output, you must always identify the portion of the entire database that is of interest. As explained in the concepts and definitions section, this is done by specifying a SelSet that defines the specific elements (usually geographical areas) that are to be included. Any number of different SelSets may be defined for a given winR+ database and each Set can be built from any combination of areas identified in the data. Of course, you can also select the entire area, i.e., all the database.

In this exercise, you will produce and display frequencies and cross tabulations of education-related variables using data from the national census taken in 1990 for two areas of interest:
a) The entire Puerto Nuevo county; and

b) A planning area within the county of Puerto Nuevo.

Figure 11. Tutor2a Selection Set for Tutorial-2 created in the Selection Editor. The SelSet comprises Puerto Nuevo of the county entity. The line at the bottom shows the hierarchy up to the entity level of the area highlighted.


See the text for more information.

Creating a simple Selection Set

You first want to obtain results for the entire county of Puerto Nuevo. For this, you want to create the SelSet Tutor2a

Procedure

1. Start with a Workspace open. As always in winR+, you must have a current Workspace open—if you have not done so, you should open the New Miranda Workspace—.

2. Create a New Selection Set. Click the New icon on the Tool bar, or select the document type Selection Set in the Workspace and click on the New button, either of which opens a Selection Editor window; this will initially show only the root entity wmir with an empty circle to the left. Refer to Figure 11 during the following steps.
Chapter 4. A walk through winR+: [Tutorials] — 41

3. Expanding and selecting entity elements:

3a **Expand a geographical area**: With a click, highlight the area name that you want expanded — here *wmir* —, and then click the **Expand** button of the Selection Editor. The next lower level of elements or members of the entity will appear (see Figure 12 for a more expanded tree than in Figure 11).

3b **Select/deselect an entity element**, in this case, *Puerto Nuevo*.
- Highlight *Puerto Nuevo* by clicking on it with the left mouse button.
- Select the current area by clicking the **Select/Deselect** button. The empty circle to the left of the name will turn fully Grey indicating a *TOTAL* selection of Puerto Nuevo. The circle for the entire WMIR country will become half-Grey, indicating a *PARTIAL* selection, since only part of it has been selected.
- If you have made a mistake or otherwise wish to deselect a highlighted area, click the **Select/Deselect** button again; the circle becomes empty again. The SelSet should appear as Figure 11.

*If you had this selection made and then you highlighted a higher level entity, in this case WMIR, and tried to deselect the previous selection with the Select/Deselect button, you would Select all the lower elements. Then, if you tried to correct this using the same button again, you would deselect everything and you would have lost your previous selection! To avoid such an error, particularly when the selection is complex, it is recommended that you save your work frequently.*

4. Save the Selection Set:

- **Use the Save icon or File|Save** on the Main Menu to save the SelSet to your Workspace.

- **Enter a name of up to 8 characters when saving the new SelSet**, or when doing a **SAVE AS**. You should replace the default name of “UNTITLED” if it appears; *Tutor2a* is suggested in this case. You can also add a descriptive label of up to 40 characters. When saved, the SelSet appears in the Workspace list.
Creating a Selection Set in a multi-branch database

The second SelectSet for this Tutorial is a planning area within the county of Puerto Nuevo; the area comprises all of the district of San Antonio plus the 90area entity = 1 of the district of Concepción. The final result of the selection is shown in Figure 12 (see the caption of Figure 11 for the meanings of the circles). The problem here is that the hierarchy branches as you move down it, so you must tell winR+ which branch to follow.

Procedure

1. Follow the steps 1-3 in the previous procedure. As in the previous selection, expand WMIR to the county level.

2. Expanding and selecting from a multibranch.
   2a Choosing a branch: Now you must expand Puerto Nuevo to select parts of it. However, as you can see from the database structure (open the Dictionary by clicking on its icon or look at the printed Dictionary window, page 38), the WMIR database has various branches and winR+ has no way of knowing the branch that you want to follow, which in this case, is the branch continuing with the entity district.
   - When you try to expand an entity which branches at the following level down, winR+ requests information concerning which of the branches to expand, by opening the window for you to make a selection from a combo box.
   - After you have expanded the combo box and clicked on your selection of the next entity (district in this case), click on the Done button.

   2b Selecting All of an area: You can now select all of San Antonio with the Select/Deselect button (the San Antonio circle turns black, and the circles of Puerto Nuevo and WMIR become half-black because a part of each has been selected).

   2c Selecting part of an area: To select the 90area entity element = 1, which is within Concepción, you must expand
Chapter 4. A walk through winR+: [Tutorials] — 43

Concepción; and then pick the next entity on the branch you wish to continue, which in this case is 90area. After the 1 and 2 codes appear for 90area, highlight the area coded 1 and select it with the Select/Deselect button.

3. Save the Selection Set: Make a Save and name it Tutor2b with a label like: "Tutorial: Puerto Nuevo + (90area=1)".

If you have made a mistake or otherwise wish to contract to a higher level, for example, contracting the elements of 90area to return to the district=Concepción level, click on Concepción to highlight it and then on the Contract button.

Note that the areas in a SelSet can be administratively or geographically disparate, e.g., areas in the set may come from two different provinces. Hence, any new area for which information is required can be created from any combination of the original areas in the data.

Figure 12. Tutor2b SelSet for a planning area consisting of the entire San Antonio district plus the 90area entity =1 of the Puerto Nuevo district of Concepción (partial view of the Selection Editor).  
{siSEL2b.wpg}

An alternative method: mouse clicking a Selection Set

- **Highlight an area**: Left click on its name.

- **Expand an area**: Double click a highlighted area (equivalent to highlighting the area and using the Expand button).

- **Contract (or un-expand) an area**: At the level to which you wish to contract, left click to the left of the circle (equivalent to highlighting the higher level and using the Contract button).
- **Select an area:** Right click a highlighted area (equivalent to highlighting an area and then using the Select/Deselect button — Deselect and Unselect are used interchangeably).

- **Deselect an area:** Highlight the selected area and right click it (equivalent to highlighting and using the Select/Deselect button).

*winR* provides the means to create a Selection Set when you have quantitative criteria that may require the prior calculation of indicators using the winR+ database or external information; see page 177. In addition, winR+ also permits you to click on areas of a map display to create a Selection Set; see page 194.

**Writing commands with a mouse: EasyTabs**

Based on the data in the 1990 Nueva Miranda census and for each of your two selection areas, *Tutor2a* and *Tutor2b*, you will now ascertain the frequency distribution of the population for the variables describing school attendance, type of education and sex, and you will make a cross-tabulation of type of education by sex to see how the type varies by gender.

You may wish to refer to the dictionary (clicking on its icon) to see whether the relevant variables are there. In the dictionary you will find that each variable has a mandatory short and often cryptic name of up to 8 characters and a longer more descriptive name (often called a variable *label*). To find the variables in the Dictionary, click on the 1990 person entity, *90person*, and then look down its list of variables to find, respectively, *90attend*, *90edtype*, and *90sex*.

To tell *winR* what you want to do with the variables, you normally must write a Command Set to describe the frequencies, cross tabulations and averages that you require. But to create output tables that do not require the creation of any new variables, you can use the *EasyTabs* facilities of *winR*. With *EasyTabs* you do not have to write any commands at all — or even know anything about them—.
When EasyTabs can be used

*EasyTabs can be used to produce frequencies, cross tabulations, averages and simple counts. In each of these, original dictionary variables, and only original variables, can be used; that is, those that are a permanent part of the dictionary and physically exist on the associated database. EasyTabs cannot be used with "new" variables that you derive from the original variables.*

Frequency distributions with EasyTabs

Procedure

1. **Open the Easy Frequency window:** A click on the Main Menu Tools|Easy Frequency brings up the Easy Frequency window seen in Figure 13. The window with the Frequency Tab will visible (with the Report Tab window covered); if it is not, click on the Frequency Tab.

![Easy Frequency window](image)

2. **Choose a Selection Set:** Choose the area of interest by clicking on the down arrow in the combo box in the upper-left corner of the Window and then select SelSet Tutor2a. If you do not have this
ready, create the SelSet using the Selection Editor as explained on page 39.

Since only one SelSet can be used at a time, you must repeat the entire procedure after completing the first set of frequencies to run them for Tutor2b.

Note that winR+ comes with the ALL SelSet, which is always available if you want to process the entire database.

3. **Choose variables:**

- **Choose the entity level of a variable.** A variable always applies to an entity. Therefore, to tell winR+ which variables you wish to process, you must first choose the entity of interest. In this case, you want the 90person entity which can be chosen from the entity list in the combo box on the upper left of the Frequency Tab window in **Figure 13**. Click on the little down arrow to open the list and then click on the 90person entity.

- **Select the variables** of interest. Then in the variable list box, click consecutively on each of the three variables of interest, 90attend, 90edtype, and 90sex (you may have to scroll the list). Their full entity.variable names will appear in the box on the right.

**Deselection:** If you make a mistake, you can deselect a variable that appears in the right-hand box, by double clicking it.

- **Select Options**, as you wish:

  **Include missing values:** Leave this box blank — the default — when you want the missing values excluded from the table, i.e., neither included in the totals or the percentages, if any.

  **Include zeros:** Leave this blank — the default — when you do not want rows with zero cases to print. If this is checked, all rows appear within the range. This latter presentation may be convenient when comparing two variables with the same categories in order that each have the same number of rows.

**Save the Command Set.** The Easy Frequency window internally generates the instructions to carry out the process; these can be saved as a Command Set by clicking in the Save Command Set on Exit box. To avoid cluttering up your Workspace, you should not save Easy Frequency or and other EasyTabs Command Sets unless you have a real use for the them.
You can see the Command Set without clicking this box, by clicking on the **Report Tab** after running you run the process and clicking on the underlined word **List**. See page 54 for more on Report.

- **Run the process.** Click on the **Run** button at the top of the window.

4. **View the Output** (no action required, since the three frequencies tables will appear automatically in individual windows, one of which is shown in **Figure 14**).

You can close any or all the output tables individually. To recall any of the outputs, click the **Report Tab** and then click on the underlined outputs of interest. See page 54 for more on Reports.

5. **Print the Output.** You can print this and any other **EasyTabs** outputs by clicking on **File|Print** on the Main Menu, accepting the default printing options.

   **Print Preview:** If you want to see your page first, click on **File|Print Preview**.

   **Page layout:** You can change a large variety of layout options, including margins, headers, footers, etc., by clicking on **File|Page Setup**... (see **Figure 24** on page 71). However, the printing font is changed by clicking on **Tools|Preferences** and then clicking on the **Printing** Tab.

6. **Graph the output.** You can graph any Frequency as seen in **Figure 14** (as also any Average). Click on **View|Graph Presentation** on the main menu. If you want to customize the graph, you can click on **View|Options** to select a wide variety of graph types, styles, colors, etc.

7. **(if relevant) Make a Save:** If you marked the **Save Command File** on **Exit** box and want to save it to your Workspace, click on the **Exit** button of the window when you are ready to permanently close your **Easy Frequency** window. A window appears asking for a name and a label.

As you did for the SelSets (page 41), give the Command Set a name of
up to 8 characters and an optional label. If you want to save the CmdSet(s) and did not check the Save... box when defining the tables, you must start again so that you can check the box.

Cross-Tabulations with EasyTabs

Now that you know how the work with the Easy Frequency window, you can quickly become familiar with the other EasyTabs windows: CrossTabs, Averages and Simple Count. The exercises can be done for both the Tutor2a and Tutor2b Selection Areas for practice, although in the text only the former will be used. As examples of CrossTabs, we shall make the following two cross tabulations of dictionary variables:

90person.90attend by 90person.90sex and
90person.90edtype by 90person.90sex.

Figure 14. Output window for one of the three Easy Frequency tables obtained in Tutorial 2. A bar graph is shown with cross-hatching for monochromatic printing. (siEasyFO.wg)
Figure 15. The EasyTabs window for requesting CrossTabs of original dictionary variables without having to write a Command Set. {siEasyC.wpg}

Procedure

1. **Open the Easy Crosstabs window**: Click on the Main Menu Tools|Easy Crosstabs to bring up the window seen in Figure 15. The window with the Crosstabs Tab should be visible.

2. **Choose Selection Set...Run the process**: Follow the outline for making an Easy Frequency (starting on page 45). As there, in Easy Crosstabs, the variables are copied from the entity and variable name boxes on the left to the row box by clicking first in the latter and then clicking the entity and variable(s) of interest; the same is done for the column box. Note that winR+ produces a crosstabulation output for each combination of row and column variables.

   **Percentages**: If you want one or more percentages, click the appropriate boxes.

   The row and column variables refer to the fact that the categories/codes of the variables will be on the left row stub and the top column stub, respectively. In turn, the row percentage means that the
sum of the percentages in a row will be 100% and similarly for the column percentages.

3. View and Print the Output...Make a Save: These actions are also essentially the same as those done for EasyTabs Frequency. The requested results appear in the Output windows, one of which is shown in Figure 16.

4. Graph the output. In the first release of winR+, no graphs are available for Cross tabulations.

Notice that the percentages given in Easy Crosstabs output of 90person.90attend BY 90person.90.sex are meaningless for many purposes, since the 'Never attended' category may include relevant school age or older persons who never attended as well as babies and other very young children who are not eligible to go to school—or these may be in the "no response" category, confusing the percentages. In the first release of winR+, this category or groups to which the tabulation does not apply (e.g., children under school age) cannot be filtered in EasyTabs; later versions may allow filtering. In any case, the winR+ language solves this problem (see page 72).

![Figure 16. Output of Easy Crosstabs.](siEasyCO.wpg)
Averages with EasyTabs

You can also make an EasyTabs Average. In this case, assume you want to determine the average age of persons living in houses according to whether their houses have water or not and the type of walls, that is, you want the average of:

\[ \text{90person.90age} \text{ BY } \text{90housin.90water} \text{ BY } \text{90housin.90walls}. \]

Procedure

1. Open the Easy Average window: On the Main Menu click on Tools\n   Easy Averages (Figure not shown).

2. Choose Selection Set...Make the Save: These actions are essentially the same as for Easy Frequency (see page 45).

Note that in the example for this Tutorial you are working with two different entity levels, namely, persons and houses, \textit{90person} and \textit{90housin}, respectively, in the New Miranda database.

\[ \text{This is an example of winR+ hierarchical processing when the entities involved all are on the same branch. You must learn how to interpret your results — winR+ takes care of the processing automatically—. More complex examples of hierarchical processing are given in Tutorial 4 and in Chapter 7.} \]

Counting things in sub-areas with Simple Count

When developing a health plan for a local clinic, you may need to know, for example, how many male children 0-4 years old live in each block of your planning area. In New Miranda, this requires that winR+ count the number of persons of the \textit{90person} entity who are within each of the blocks of the \textit{90block} entity within the planning area, subject to the condition that the persons are males and less than 5 years old. For uncomplicated problems as this and that use only existing dictionary variables, winR+ provides Simple Count; like the other EasyTabs, it does not require any knowledge of the winR+ command language.
The result of a Simple Count is a table, such as that shown in Figure 17, listing the code of each sub-area of interest and the number resulting from the count for that sub-area. Thus, the first block shown, 902101009, has 12 children meeting the condition of being male and less than 5 years old. This type of table is called an AREALIST in winR+. As many geographical analyses begin with an AREALIST—which may have various columns with information on different variables—, winR+ provides many facilities for creating and managing AREALISTS, displaying the results on maps in winR+, and for exporting and importing the results to and from other software including Geographical Information Systems (GIS). These topics are discussed in Tutorial-4 in Chapter 5 and in Chapter 7 on additional topics and procedures.
Chapter 4. A walk through winR+: [Tutorials] — 53

Figure 18. The Simple Count window set up to count the persons living on each block who are males less than 5 years old.

Procedure

1. **Open the Simple Count window.** Click on the Main Menu Tools|Simple Count to bring up the window seen in Figure 18. The first Tab called Simple Count should be on top; if not, click on it.

2. **Choose the Selection Set of interest,** *Tutor2a,* in this case.

3. **Fill in the three required boxes:**

   - **Select output name box.** In the uppermost box, provide a name under which your output will be saved in your Workspace.

   - **Select output entity box.** Indicate the entity level for the counts, in the present case, *90block;* that is, there will be a row for each of the sub-areas or elements of the *90block* entity within the Selection Set.

   - **Select entity to count box.** Indicate the entity elements that should be counted; here *90person* is to be counted. It could also be any entity lower than the output entity; thus, the number of houses on each block could also have been counted.

   - **[Optional] Fill in the condition,** if only a subset is to be counted. In this case, only males who are less than 5 years old are included. Clicking on the Condition button, brings up an Expression Builder (see page 168 for guidance on writing complex expressions). The relatively simple expression here is:
90person.90sex = 1 AND 90person.90age < 5

winR+ evaluates your expression as each “case” is considered to be counted or not; when the expression is True, the particular element (here a person) is counted. When the expression is False, the person is excluded from the count.

• [Optional] Save command set on exit box. If you want to save the Command Set made by Simple Count, check the small box above the Condition button. To see the command, click on Display Command.

4. Run the process. Click the Run Button.

If you already have an AREALIST in your Workspace with the same name—for instance, you tried to Run the Simple Count twice with the same name for the Output—you will get an error message. You may change the name of the Output or erase the existing AREALIST of the same name (highlight the name in the Workspace and click the Delete button).

5. View your results. At the end of the Run, the AREALIST appears automatically. If you have closed this output, you can call it up again from the Report if it is still open, or at any time in the future by opening the AREALIST from your Workspace (see page 34). See the next section for the use of the Report.

Viewing results via the Output Report

The Report window: Re-process the Easy Frequency exercise again (or Crosstabs, Average or Count) and then click on the Easy Frequency window so that it becomes the active window. Click on the Report Tab. You will see that there are underlined phrases in green; these are hypertext links, which if clicked upon, will each open a window with the information alluded to (see Figure 19).

List and Tables. These use hypertext links to connect the Report to the actual outputs.

• The list of Tables: Clicking on an underlined green hypertext link of Table will bring up the respective Output Window, if it has been closed. This is a general approach that is used throughout winR+ to manage what is often a large number of output tables from a single process.
• The **List of the Command Set**. Click on the hypertext link List to see the Command Set that was created for you to obtain the output. In the next Tutorial you will learn to create your own Command Sets.

![Image of Easy Frequency window]

**Figure 19.** The *Report* window from an Easy Frequency. Clicking on the hypertext links—underlined words, in green on a color monitor—brings up the respective information.
Chapter 5.

A WALK THRU winR+:
SPEAKING THE LANGUAGE [Tutorials 3-4]

If you read the previous chapter and that on the basic concepts of winR+ (starting on page 21), you have become sufficiently familiar with winR+ to navigate among its windows, use your own Workspace and produce simple tables. But you may have become frustrated by the limitations of working without a language to express to winR+ what you want to do with your data and what outputs you want.
TUTORIAL-3: The winR+ language

This tutorial introduces you to the winR+ command language. In particular, you will learn how to:

- Identify variables by their entity variable names;
- Structure Command Sets based on three basic winR+ commands.
- Get assistance to recall and write commands and variables;
- Use the three basic commands RUNDEF, DEFINE and TABLE;
- Modify the basic commands with the keywords UNIVERSE, FOR, RECODE and OPTIONS; and
- Create new variables at a higher level (hierarchical processing).

If you require more detailed information on the use of a given command keyword, refer to the Reference Guide to the winR+ Command Language in Appendix B (starting on page 217).

Identification of variables in winR+

As you know from the chapter on basic winR+ concepts (page 21), any entity can have a set of variables that describe its elements. This is equally true for entities that are selectable geographical areas like districts as it is for nonselectable entities (see page 23) like houses and persons — e.g., the entities 90housin and 90person, respectively, in New Miranda—. A variable must be identified by both its entity name and its own short name. Only if all short variable names were unique in database, would the entity be superfluous. Since this cannot be assumed for reasons that are explained in the next paragraph, each and every variable in a winR+ Command Set (CmdSet) must always be identified in the following form:

Entity. Variable

Thus, the variable for age of persons in the 1990 New Miranda census must be written 90person.90age, rather than simply 90age.

Compared with R+

In fact the "90" prefix of the variable 90age is not really necessary in winR+, since the entity 90person distinguishes the 1990 census from the 1980 census in the winR+ New
Chapter 5. A walk through winR+: [Tutorials] — 59

Miranda database. The variable prefixes are a leftover from the Redatam-Plus requirement that all variables in a database must have unique names.

The winR+ software uses the entity.variable convention to facilitate “openness”, that is, the ability of winR+ to process external databases along with a winR+ database. It is impossible to suppose that all the variable names in two or more external databases, as well as a winR+ database, will be unique in any given process. The entity.variable convention solves that problem — of course, the entity names must be unique across all involved in a process — and also facilitates the “reading” of a CmdSet by its creator and others who may wish to understand it.

Since winR+ has Assist facilities that help find and write the full variable identifications, the convention involves little extra work and has important advantages over an approach requiring unique variable names. A latter release of winR+ may provide a means of avoiding the need to write the entity except when there is a possibility of ambiguity.

In the text, variable name is often used in place of Entity.Variable name unless there is ambiguity; if the name without the entity is required, it is called the variable short varlabel — in small letters to distinguish it from the command keyword VARLABEL which is always shown in capital letters —.

Introduction to the winR+ command language [Tutorial 3a]

The winR+ command language enables you to speak to the software via Command Sets (CmdSets) that tell winR+ how you want it to carry out simple processes similar to EasyTabs (see page 44), as well as the creation of complex new variables and the production of elaborated outputs.

Compared with R+

Although initially deceptively simple, the new language is very powerful and can do everything that was done in Redatam-Plus and much more. A few commands like CASES and SAMPLE in R+ are not in the first release of winR+, but will be included if there is a demand for them. Note that a conversion program does not come with winR+ to convert old R+ programs to the new language.

The three basic commands: RUNDEF, DEFINE & TABLE

The winR+ language has only three basic commands:

RUNDEF defines the environment in which a process will occur during a Run, including the mandatory Selection Set (SelSet) that identifies the geographical area to be processed and optional conditions such as the universe to which all
processing during the Run will be limited—e.g., only females 15 to 45—, etc. This is the only required command of a CmdSet.

**DEFINE**

Creates new variables, if any, specifies their characteristics, and allows them to be saved, if desired. Various DEFINEs can be used in a single CmdSet.

**TABLE**

Describes a specific output and conditions, if any, on its content. Various TABLEs can be used in a single CmdSet.

<table>
<thead>
<tr>
<th>Example of Command Set</th>
<th>Structure of Command Set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RUNDEF</strong> 90hounit tippers</td>
<td></td>
</tr>
<tr>
<td><strong>RUNDEF</strong> 90hounit Sexed</td>
<td></td>
</tr>
<tr>
<td><strong>DEFINE</strong> 90hounit tippers</td>
<td></td>
</tr>
<tr>
<td><strong>DEFINE</strong> 90hounit Sexed</td>
<td></td>
</tr>
<tr>
<td><strong>TABLE</strong> 90hounit tippers by 90hounit Sexed</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 20.** A Command Set example and the structure of a CmdSets

### Assembling commands and keywords in a Command Set

Figure 20 shows an example of how these three commands are put together in a Command Set that winR+ can understand. On the left of the Figure is an actual CmdSet—it is not important now that you understand what it does;—this CmdSet will be explained in a later Tutorial—and on the right is a diagram showing the general components of a CmdSet. RUNDEF is always first and never repeated in the same CmdSet. The DEFINE and TABLE commands are optional, but at least one is
necessary if you want to do anything useful in a given CmdSet. As you can see in the CmdSet, there can be more than one DEFINE command and (not shown), more than one TABLE.

Each of the three basic instructions can be qualified by subordinate clauses with command keywords, or keywords, for short. Thus, RUNDEF can have clauses using the following keywords along with the information given in the right column, for example, UNIVERSE

90person.90sex = 2:

RUNDEF
RUNTITLE
SELECTION SET
UNIVERSE

Short identification of the Run.
Title in quotation marks.
Area to be processed.
Expression indicating the universe to be included in the Run.

Similarly, to DEFINE a new variable at any entity level, you can use:

DEFINE
AS
FOR

States the new variable name.
Expression that defines the new variable.
Expression filtering, i.e., limiting, the cases for which the DEFINE applies (similar to an “if” in Redatam-Plus).

TYPE
RANGE
VARLABEL
VALUELABEL

Whether the variable is an integer or real (i.e., has decimal values).
The lowest and highest values of the variable.
Descriptive name of the new variable.
Labels individual categories (codes) of the variable.

Some of these, in turn, may be further modified, such as to DEFINE a variable AS the RECODE of another variable, etc.

And finally, for the TABLE command, the output is specified using:

TABLE
AS
OF
FOR
OPTIONS.

Short table identification name,
FREQUENCY, AVERAGE, CROSSTABS,
or AREALIST.
The variables in the output.
Expression filtering or limiting the cases to be included in the output.
To specify details about the form of the output.
In addition, there are other subordinate clauses and modifiers that further enable you to manipulate your data and produce desired outputs. All will be treated in the tutorials that follow and in the succeeding chapters.

Using the Reference Guide to the Command Keywords

Tutorials 3 and 4 in this Chapter are designed to help you begin to work with the winR+ language and to use the winR+ facilities for writing CmdSets. However, for looking up details about a particular keyword, you will probably find the Reference Guide to the winR+ Command Language in Appendix B (starting on page 217) much more rapid and complete. It is organized alphabetically by keywords within the RUN-DEF, DEFINE and TABLE commands.

Getting an Assist

To help you write Commands Sets, there are winR+ Assist windows, similar in function to Wizards in other Windows software. The Guided Assists help you create specific RUNDEF, DEFINE and TABLE commands in your CmdSet, providing you with lists of the keywords, your SelSets and the entity.variable names. All these are selected with the mouse so that you have to write very little, if anything; after automatically checking that the syntax (grammar) is correct, the full command is transferred to your CmdSet. Should you want to make a change, you can transfer the command back to the Assist for additional help, or make the changes directly in the Command Editor.

If you prefer to write the commands directly, a General Command Assist provides you with all the entity.variable names and command keywords that can be dragged and dropped from the Assist to your CmdSet in the Command Editor window.

A simple CmdSet with RUNDEF and TABLE [Tutorial 3a]

The first exercise in this Tutorial is to create a CmdSet to obtain in one Run the same frequencies, crosstabulations and average produced in the three separate EasyTabs processes of Tutorial-2. The CmdSet that you will create is shown in the Command Editor window in Figure 21.
Chapter 5. A walk through \textit{winR+}: [Tutorials] — 63

Figure 21. The Command Editor window; shows the CmdSet for the first exercise of Tutorial-3a.

\textit{Procedure}

1. \textbf{Open the Command Editor}: To open the Command Editor window, go to the Workspace window and select the document type \textit{Command Set} or, alternatively, click the CmdSet icon \textbullet. Then click the New button to open the Command Editor window. The window shown in \textbf{Figure 21} will appear; initially it will be blank.

\textbf{Save As}: It is recommended that you make an initial \textit{Save} now so that you can make occasional \textit{Saves} as you work. Click File|Save As on the main menu and give the CmdSet its mandatory short name, such as \textit{T3a}, and an optional descriptive label, such as, “Tutorial3a: CmdSet to reproduce outputs of Tutorial2”. Click occasionally on File|Save when making future \textit{Saves} of the CmdSet, as you work.

2. \textbf{RUNDEF, always the first command}: A CmdSet must start with a \textit{RUNDEF} command which specifies the SelSet and other general parameters, as needed. It is \textit{always} the first command and appears only once in a CmdSet. To facilitate the writing of the command, you can use a \textit{Guided Assist} window:
Users who know the language well may often find it easier and faster to work directly in the Command Editor window, using the General Command Assist to drag and drop SelSet, entity and variable names and keywords. The General Command Assist can be accessed by the Assist button at the top right side of the Command Editor window (see Figure 28 on page 79).

A Guided Assist for making a RUNDEF command will be used in the present exercise to show how to use a Guided Assist — there are also Guided Assists for the DEFINE and TABLE commands.

2a. **Access the “Guided” RUNDEF Command Assist:** After opening the Command Editor, you can obtain step-by-step assistance with specific commands by clicking on the right button of the mouse (click once in Windows 3.x and a double click in Windows 95) or click Tools|RUNDEF Assist (note that the Command Editor must be the active window when you use the Tools menu for this purpose). The window shown in Figure 22 will appear.

2b. **Fill-in the blanks in the RUNDEF Assist:**

- **Complete the Job ID**, which must not have any blanks spaces.

- **Complete the optional Job-Title** as shown in the Figure.

- **Indicate the SelSet** that defines the area that you wish to process. The Type box should indicate SELSET. Then choose a SelSet from the Name combo box showing all the SelSets (Selection Sets) in your Workspace; in this example, choose Tutor2a.

  If you also want to use Tutor2b, you must create a separate CmdSet (or simply edit the present CmdSet after processing the completed first SelSet; if you save the second CmdSet, use SAVE AS to give it another name).

- **Display the command** in the RUNDEF Assist by clicking on the Display command button at the top left of the window. A window will appear with the command written as it will be in the CmdSet (see Figure 22). If you want to make changes, click OK to return to the Assist. — For the moment,
you do not need to use the universe box at the bottom of the Assist window—.

- **Transfer your work to the Command Editor** by clicking the Done button at the top of the RUNDEF Assist window.

- **Check the syntax.** In the Command Editor you can determine whether you have any "grammatical" errors by clicking the Compile button. If there is an error, winR+ will indicate where and give an idea of what the problem might be.

After using this guided Assist (or the others for DEFINE and TABLES) to place a command in the Command Editor, you can make changes directly in the Editor, or your can return to the guided Assist if you want to make changes or additions. This is done by highlighting or "blocking" the entire command and then clicking the right button of the mouse once in Windows 3.x and twice in Windows 95 to select the appropriate Guided Assist, here RUNDEF. This can be done even if you wrote the command yourself the first time. However, although the command need not be complete, note that the syntax must be correct, or it will not be accepted by the Guided Assist.

3. **Specify the TABLE(s).** In general, each specific output requires a separate TABLE command. Thus, there will be a TABLE command for each set of FREQUENCY, CROSSTABS and AVERAGE. We shall use the Guided Assists for two types of TABLE.
3a Indicate the TABLE location: With a mouse click, put the cursor in the Command Editor in the line after the end of the RUNDEF command, where you want your first TABLE command, since when you finish with the TABLE Assist window (see the next step of the Procedure), the results will be sent directly to the Editor and to the place where the cursor is — if you want, you can skip a line after the end of the RUNDEF command for readability —

3b Use the TABLE Assist: With the Editor window active, i.e., the top banner of the Windows is colored, click on Tools|Table Assist in the Main Menu or click on the right mouse button, once in Windows 3.1 and twice in Windows 95. You will make the first TABLE as a FREQUENCY.

- Fill in the TABLE Tab of the TABLE Assist Window shown in Figure 23. The Type [of TABLE] combo box should indicate Frequency.

Then give the TABLE a Name, e.g., “Tut3a_Frequencies” and, optionally, a Title. The “_” in the Name makes the Name a single word because it avoids a blank. Leave the FOR and AREABREAK empty since the frequencies required in this example have no modifiers; the Options
boxes will also be left unchecked since we do not want values that have zero cases nor do we want to include missing values (i.e., values outside the defined Range of values, which should be defined when the database is created).

![Table Assist](image)

**Figure 23.** The Guided Table Assist for creating a Frequency TABLE, showing the window when the Table Tab is clicked. (siFA3a.wpg)

- **Fill in the FREQUENCY Tab:** This tab appears because you indicated that the TABLE Type is a Frequency. Click on the FREQUENCY Tab to change the window, if it is not on top. It similar in operation to the Easy Frequency window that you used on page 45. The variables that you want are:
  90person.90attend
  90person.90edtype
  90person.90sex

- **Transfer the TABLE to the Command Editor:** After checking the command by clicking on the Display command button, click the Done Button. Your entries,
which are “pasted” in the Command Editor, should appear similar to:

**TABLE** Tut-3a: Frequencies
AS FREQUENCY
OF 90person.90attend, 90person.90edtype, 90person.90sex
OPTIONS "Tut-3a: Simple Freq of Tut2"

Note that there is NO continuation character in winR+ to continue from one line to another. As long as the keywords are used correctly, the software knows when a line is continued. Furthermore, since the continued line(s) need not continue from the margin, you can format a command for easy readability with spaces or the **TAB** key.

As you can see, the language is relatively close to natural language. The third line of this **TABLE** command can be made more readable by widening the Editor window so that the line appears as:

**OF 90person.90attend, 90person.90edtype, 90person.90sex**

The window size has no effect on the outcome, but widening the window will make it more readable.

*It is strongly recommended that you indent subordinate clauses of each command and that you use upper case letters for Command keywords and lower case letters for variables and other text. These will facilitate easy reading by visually clarifying the structure of the commands.*

4. **Use the TABLE Assist for CROSSTABS:** Open the **TABLE Assist** (not shown) again, fill-in the **Name** and **Title** boxes and select **CROSSTABS** from the **Type** combo box.

- **Define the Crosstab.** The **CROSSTABS** Tab window is similar in operation to the **Easy CROSSTABS** (see page 49). Enter the information for:

  90person.90attend BY 90person.90sex and
  90person.90edtype BY 90person.90sex.

*Note that both tabulations can be entered together because 90sex is common to both. If you also wanted a third tabulation that had totally different variables, you would have to request them in a separate **TABLE** command.*
You will see that the commands for CROSSTABS (and AVERAGE) are written similarly to the FREQUENCY command, that is,

```
TABLE <TableID>
AS CROSSTABS
OF variablen1 BY variablen2
```

Up to three BY can be used to get a 4-way crosstabulation.

- **Indicate any Options.** After completing the CROSSTABS Tab, return to the TABLE Tab to select the options that you want: select column percentages so that you can compare males and females —column percentages have the 100% at the bottom of each column and row percentages at the end of each row—.

- **Transfer the TABLE to the Command Editor** by clicking the Done button. If you forgot to filling in any required box, you will get a message. After completing the missing information, again click on Done.

5. **Use the TABLE Assist for AVERAGE.** Now open the TABLE Assist and select AVERAGE in the Type box. The window for AVERAGE will appear. It is similar in operation to the EasyTabs AVERAGE (page 51) and the CROSSTABS Assist, just completed. You should enter:

```
90person.90age BY 90housin.90water BY 90housin.90walls
```

The first variable is the one that is averaged and therefore should normally be a quantitative variable like 90age. Up to three BY can be used in an AVERAGE. After completing the information in the two windows of the Assist and checking the command, click Done.

6. **SAVE and RUN.** As noted earlier, the final CmdSet should be essentially the same as in Figure 21 on page 63; the List is also shown in 72 along with the Statistical Process Report.

- **Compile the CmdSet.** Use the Compile button in the Command Editor window or the Compile icon to check the syntax. If there are compile errors, you must correct them to be able to make a Run.
• **Run the CmdSet.** Click the **Run** button or its icon \[\text{Run}\]. If you wish to stop the process before it finishes, click on the **Abort** icon \[\text{Abort}\]. Note that *winR+* always makes its own compile before doing a run, so any errors will be found even if you do not make a Compile first.

7. **View and print the results:** After the Run, the Command Editor *winR+* Statistical Process Report will appear (see Figure 25). Like the output report of EasyTabs (see page 55), you can access any of the outputs by clicking on the green underlined hypertext links. Any of the Output windows can be printed by clicking on **File|Print** on the main menu.

8. **Customizing the look of the printed output page.**

• **Go to the Page Setup window** to customize how a Table or CmdSet List will appear on the printed page. With Output window containing a Table or a List must be active, click on **File|Page Setup** to bring up the Page Setup window for formatting your output. Some of the adjustments that you can make are:

  **Header and Footer:** You may add any text to each of these. In addition there are codes for obtaining the date, page number, etc. These include:
  \[&D = \text{date} \quad &T = \text{Time} \quad &P = \text{Page no.} \quad &F = \text{sheet no.}\]

  **Margins:** inches or centimeters, depending on the Units box setting. The Print Setup — see the next point — fixes the page orientation.

  **Center** the output horizontally, vertically.

  **Grid lines:** Spread-sheet like grid of lines; normally off.

  **Row/Column headings:** Assumes a spreadsheet for the entire page (except the header and footer) and labels the Rows with numbers and the columns with letters, as is usual with spreadsheet displays on a monitor.

  **Scale:** If the **Fit to Pages(s)** box is checked, the entire table or list is made to fit on a single page (or the pages width and pages height indicated in the following boxes), but using a font of the
appropriate size (which may be very small if the table is very large).

Figure 24. Page setup to customize printing of a table. (siPrPg.wpg)

- Go to the *Print Setup window* by clicking on File|Print Setup on the Main Menu. You normally will only want to change the following:

  **Orientation:** Select whether you want the page to print vertically (portrait) or horizontally (landscape) depending on the shape of your table.

- **Changing the output font:** *winR* allows you to change the output font for printing. While in an Output window, click on Tools|Preferences on the Main Menu. A window will open with various Options. Click on the Printing Tab and then on the Select button, which brings up a complete list of fonts. Select a font, font style and size. Click OK and then Exit and Save.

- **Preview before printing** by clicking on File|Print Preview on the Main Menu.
Figure 25. Command Editor Statistical Process Report and a List of the Command Set used for Tutorial 3a.

Excluding ("filtering") cases with UNIVERSE and FOR

You may remember that outputs from the EasyTabs often are of limited use because dictionary variables must be used "as is"; ineligible cases cannot be excluded (see page 50). The full command language has a very easy solution to this problem via two subordinate clauses: a) UNIVERSE with RUNDEF when the restriction applies to all the outputs of a Run, and b) FOR with TABLE when the restriction applies only to that output. Note that FOR can also be used locally when you DEFINE a new variable — see the next exercise of this Tutorial —.

UNIVERSE: To filter the cases of an entire Run. Return to the Command Editor to edit your command file; if necessary, Open the CmdSet again (called Tut3a here). You can make the changes in the Editor, or by highlighting the command in the Editor and calling up the appropriate Guided Assist. Add a UNIVERSE clause to the RUNDEF command so that it reads:
Chapter 5. A walk through winR+: [Tutorials] — 73

RUNDEF Tutor3a
RUNTITLE "Tut3a: Simple Freq, CROSSTABS, Average"
SELECTION tutor2a
UNIVERSE 90person.90age >= 5

That is, the "universe" of persons included in the entire Run or job are now those with an age "greater than or equal to" 5 years, supposing that 5 is the age of school entry in New Miranda. As each case within the SelSet enters the winR+ processor, the expression 90person.90age >= 5 is evaluated and when it is True, the case is included in the "universe" of the Run. In effect, the data has been filtered to allow processing only of cases meeting the True condition. Since UNIVERSE is used in the first command before any new variables have been DEFINEd, of necessity, all variables in the UNIVERSE expression must come from the Dictionary.

![Table 5](image)

**Output: TABLE 5**

<table>
<thead>
<tr>
<th>&amp; #220;ntent</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attending</td>
<td>631</td>
<td>631</td>
</tr>
<tr>
<td>Attended</td>
<td>1879</td>
<td>1879</td>
</tr>
<tr>
<td>Never Attended</td>
<td>175</td>
<td>175</td>
</tr>
<tr>
<td>Total</td>
<td>2865</td>
<td>2865</td>
</tr>
</tbody>
</table>

Figure 26. Output of TABLE AS CROSSTABS, with UNIVERSE 90person.90age >=5 in the RUNDEF command to eliminate children too young to attend school.

All the output TABLEs will be affected. For example, looking at the results of the crosstabulation (see Figure 26) and comparing it with the Figure on page 50, it appears that the children not yet of school age were
included as "No response" since this category now has zero persons in it and the percentages are therefore more useful.

**FOR: To filter the cases processed by a single command.** The fact that UNIVERSE limits the cases for all aspects of an entire Run can be undesirable, as is the case of the AVERAGE included in the same Run of the CROSSTABS. You may want the average age of all persons living in badly constructed houses without an internal water source, or you might want it for all working-age persons, etc. For these purposes, FOR is used as a subordinate clause of a specific command. As an example, return to the Command Editor, remove the UNIVERSE from the RUNDEF, and add the FOR to the CROSSTABS command, so that it reads:

```
TABLE tut-3a_Crosstabs
  AS CROSSTABS
  OF 90person.90attend, 90person.90edtype  BY 90person.90sex
  FOR 90person.90attend >= 5
  OPTIONS
    TITLE "Tut-3a: Simple CROSSTABS of Tut-2, with FOR"
    PCC
```

The FOR is evaluated for each case and when it is True, the person is included in this specific output; the other outputs and commands in the Run are not affected. You should get should get the same results on this CROSSTABS as with the UNIVERSE command. If you want to keep this CmdSet, use SAVE AS and give it a new name and optional label.

*If you have a Run that includes only a subset of all the cases of a SelSet and further want to limit a particular output—e.g., only persons with primary school education or higher are to be included in the entire Run, and only females in one of the tabulations—, you could use both UNIVERSE in the RUNDEF and a FOR to filter only females in the TABLE; in the TABLE, there will be the effect of both the UNIVERSE and the FOR clauses.*

*Note that you can substitute the keyword FOR in place of UNIVERSE in the RUNDEF, with the same effect. It is recommended that you always use UNIVERSE with RUNDEF and use FOR when you wish to filter individual TABLEs (and, as you will see shortly, DEFINEs).*
AREABREAK to repeat an output for each sub-area

Often when processing information for a SelSet, you may want to obtain the same information for each of the sub-areas within your SelSet; for example, when processing an entire county of New Miranda, you might want to compare the districts with each other and with the entire county. One approach would be to create a separate SelSet for each district and for the entire county, and then run each separately in a CmdSet. Fortunately, \texttt{winR+} provides a much simpler solution via the keyword

\texttt{AREABREAK <entity>}

where the \texttt{entity} identifies the sub-areas to be processed within the SelSet.

The exercise here will be to produce crosstabulations for all of Puerto Nuevo (SelSet \texttt{tutor2a}) for the same variables as used in the previous exercise, but also for each of the districts making up Puerto Nuevo. You can edit the CmdSet that you already have if you saved it to your Workspace.

Procedure

1. Include AREABREAK in the TABLE command. In the Command Editor, write AREABREAK District in the TABLE AS CROSSTABS command, so that your command now appears as:

\begin{verbatim}
TABLE tut-3a
AS CROSSTABS
OF 90person.90attend, 90person.90edtype BY
     90person.90sex
AREABREAK District
OPTIONS
TITLE "Tut-3a: Simple CROSSTABS with UNIVERSE age>=5"
     PCC
\end{verbatim}

The RUNDEF command remains the same. Note that the clause beginning with \texttt{OF} does not fit on a single line and is simply continued on the next (the indenting improves readability and has no effect on the command).

2. Using the General Assist to add AREABREAK [Optional]. Alternatively you can click on the Assist Button of the Command Editor to bring up the General Command Assist window (the middle window in Figure 28 on page 79). Placing the cursor where you want the command to appear in the Command Editor, click \texttt{AREABREAK} in the Commands of the Assist window, so that the keyword appears in
the “holding” box at the bottom, and then drag **AREABREAK** to the Command Editor. Use of the General Assist is described in more detail in the caption of Figure 28 and the associated text.

3. **De-activating tables or lines without erasing** [Optional]. You can “de-activate” **TABLES** in your existing CmdSet, so that they do not process during a Run, without erasing the commands.

   **Asterisk * for Comment.** Place an asterisk *, in front of any line with a keyword line, turns each one into a *comment*; the asterisk must be the first character on the line, but need not be in the first column. Of course, this can also be used with comments that you should use to annotate your CmdSets, particularly when they are complex.

4. **RUN**

5. **View the Output.** On completion of the Run, the Statistical Process Report window will open with the usual hypertext list of the outputs. See Figure 27. However, you will notice a small difference. In addition to an green underlined “Summary” table for the entire Selection Area of Puerto Nuevo, there is also the green dotted underlined word: **Areas**.

As there is an output Crosstabulation for each of the elements of the **District** entity, i.e., for each of the districts within Puerto Nuevo,

![Image](https://i.imgur.com/53x81.png)

**winR+** opens the small window which allows you to open one or more of the tables for individual districts by clicking on the name and the **Show** button or simply double clicking on the name. Repeat for each table you wish to see. The sum of all the **AREABREAK** tables gives the Summary table that can be accessed from the Output Report window.
Chapter 5. A walk through \textit{winR+}: [Tutorials] — 77

![Figure 27. Output report containing a TABLE AS CROSSTABS with an AREABREAK. Clicking on Summary, displays the crosstab for the entire SelSet; clicking on the dotted underlined word \textit{Areas} displays a window (see text) to select the output for each of the CROSSTABS for the sub-areas.](siORABk.wpg)

\textit{Redatam-Plus} has an \textit{AREABREAK} that permits the sub-areas to be a "group" of lower-level areas. Grouping is not available in the first release of \textit{winR+}, but a grouping feature, more powerful than in \textit{R+}, is planned for a later release.

6. Print the output by clicking on File\|Print on the Main Menu. Note that the first release of \textit{winR+} allows you to print only one AREABREAK table at a time, as is also the case for any other tables in an Output Report. This limitation will be removed in a future release of \textit{winR+}.

\textbf{Processing the entire database: Selection Set ALL}

You may occasionally wish to obtain outputs for the entire database, perhaps using AREABREAK to obtain results for each sub-area of interest or for some other purpose. You can create a SelSet with the highest level entity in the Selection Editor --- \textit{wmir} in the case of New Miranda ---, which always has a single element, namely the entire area of the \textit{winR+} database, and then use this in the SELECTION clause of the RUNDEF command.
Even simpler, you can use `SELECTION` with the keyword `ALL`. That is

```
RUNDEF id
    SELECTION ALL
```

plus any other commands for the specific outputs of interest, will process the entire area of any database you are working with.

**Defining new variables [Tutorial 3b]**

Since you now are familiar with the mechanics of the `winR+` windows, menus, `Assists`, and other facilities, from here on, we shall concentrate primarily on the use of the `winR+` language. For teaching purposes, it is best that you write the commands directly in the Command Editor —with help from the `General Assist`—. For this reason, the `Guided Assists`, which are particularly convenient for users who do not work frequently with `winR+` and need help to recall the language, will not be used in this or the remaining tutorials and examples. Note that the `DEFINE Assist` which has not been explicitly used, operates similarly to the other Guided `Assists`.

**Working with the `General Assist`**

As noted above, once you are familiar with the `winR+` command language, it is usually faster —and interrupts the thinking process less—to write the command directly into the Command Editor. The `General Command Assist`, shown in Figure 28, saves time writing keywords, SelSets, and `entity.variable` names and avoids spelling errors that will give `Compile` errors. It can be accessed from the `Assist` button on the Command Editor or by clicking `Tools|Assist`.

Highlighting any item in any combo box in the `General Assist` transfers it to the “holding box” at the bottom of the `Assist`. The last item highlighted is always the one that appears in the holding box.

When you are writing in the Command Editor and come to a place where you need an item from the `Assist`, highlight what you need so that the item appears in the holding box. Then left click the holding box and holding down the left mouse button drag the item to the Command Editor. The item will fall where, and only where, `you last left the cursor`—dragging it elsewhere in the Editor does not change
where it will fall!—. Once the item is in the Command Editor, it is just like any other text.

*It is strongly recommended that you Compile your CmdSet each time you write enough of a command to be compilable—that is it must have complete clauses, since even correctly written partial clauses will give an error.—* Constant checking with Compile makes error detection and correction much easier.

If you wish to use one of the Guided Assists, perhaps for the Expression Builder (shown in Figure 29—see the brief description in the caption; see also page 168 of Chapter 7—), you can highlight any one of your compilable RUNDEF, DEFINE or TABLE commands in the Command Editor and call up the appropriate Guided Assist with the right mouse button.

![Figure 28. Generalized Command Assist](siAssGen.wpg)

**The need to DEFINE**

Your database normally does not have the variables that you need for your output tables, but you probably have the information that you
need to create or derive the new variables. You may simply need 5-year age groups based on single-year age data or you may have to create a complex indicator integrating, say, a multitude of dictionary as well as derived variables for persons and their houses into an indicator of poverty for each city block. For these purposes, winR+ provides the DEFINE command to create new variables from existing ones, often by using a series of DEFINE commands that build upon the previous ones.

Although we shall work directly within the Command Editor, using the General Assist when necessary, note that a Guided DEFINE Assist is available and, as with the other Guided Assists, will be especially useful for persons who use winR+ infrequently and do not remember the details of the command.

In this part of the Tutorial, you would like to obtain two results for the entire country of New Miranda:

Output 1) The average age of youths (considered age 5 through 29) by school attendance status in 1990 (presently attend vs. not presently attending) by the highest type of schooling attained and by sex.

Output 2) The number of males and females in 1990, aged 15-29 by 5-year groups, who attend/attended a technical, commercial or professional institute.
Chapter 5. A walk through winR+: [Tutorials]

Figure 29. The Expression Builder window is available in each of the Guided Assists to build expressions used with keywords like FOR, UNIVERSE, AS, etc. Clicking on an entity (first box) and then a variable (2nd box) highlights each; then double clicking on the variable transfers the *entity* variable name into the expression box. The operator buttons require only a single click to be entered into the expression. If a category value of a variable (3rd box) is required, double clicking it will pop it into the expression. You can also directly write in the expression box. (siExpBld.wpg)

Compared with R+

The equivalent command file (.spc) for these outputs, written in the R+ language, is given on page 211.

The mandatory first command, RUNDEF, requires a SelSet for the whole country. As you know, winR+ provides the keyword ALL for this purpose. Since both outputs are within the range 5-29 years, a UNIVERSE clause could be used with RUNDEF. Thus, the first command would be:

**RUNDEF T3b**

```
RUNTITLE "Tutorial3b: Education and attendance"
SELECTION ALL
UNIVERSE 90person.90age >=5 AND 90person.90age <=29
```
However, rather than use `UNIVERSE` in the `RUNDEF` command, we shall use a `FOR` to filter the records when attendance status is `DEFINE`d.

**Grouping categories with `DEFINE...AS RECODE`**

Output 1) requires an `AVERAGE`

```
OF 90person.90age BY 90person.90edtype BY
    90person.90attstat BY 90person.90sex
```

where `90person.90attstat` is “School attendance status”. This uses information from the dictionary variable `90person.90attend`, but some of its categories are grouped into a new category. If you click on the Dictionary icon and then click on `90person.90attend` to see its categories, you will note that “Attended” and “Never attended” can be combined to make the category “Not presently attending”. That is, we can `DEFINE` the new variable `90person.90attstat` via a `RECODE` of the original categories of `90person.90attend` using the relationship:

<table>
<thead>
<tr>
<th>Original variable</th>
<th>New variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>90person.90attend</code></td>
<td><code>90person.90attstat</code></td>
</tr>
<tr>
<td>0  No response</td>
<td>0  No response</td>
</tr>
<tr>
<td>1  Attending</td>
<td>1  Attending</td>
</tr>
<tr>
<td>2  Attended</td>
<td>2  Not attending</td>
</tr>
<tr>
<td>3  Never attended</td>
<td></td>
</tr>
</tbody>
</table>

The `RECODE` clause of the `winR+` command for this can be written:

```
DEFINE 90person.attstat
    AS RECODE 90person.90attend
        (0  =0)
        (1  =1)
        (2-3 =2)
```
Chapter 5. A walk through winR+: [Tutorials] — 83

Note that eliminating the recodes (0=0) and (1=1) would give exactly the same result since winR+ recodes unmentioned categories into the same values, unless there is an ELSE clause — see the next paragraph for an example of this—. Note also that in this example the following forms are all equivalent:

(2-3=2), (2 TO 3=2), (2 TO HIGHEST), and (2,3=2).

Of course, where relevant, LOWEST can also be used.

The second required Output for this exercise, item 2) above, also requires a new variable. The single years of age 90person.90age, which is a dictionary variable of the Nueva Miranda census, must be grouped into 5-year age intervals. The output must also be restricted to persons who are 15 years or older (the UNIVERSE clause already has excluded persons over age 29). The 5-year grouping can be done with a DEFINE in more than one way (see page 112), but perhaps the most straightforward — and most verbose — is to create the new variable 90person.90age5 with a RECODE clause:

```c
DEFINE 90person.90age5Rec
AS RECODE 90person.90age
(15-19 =4)
(20-24 =5)
(25-29 =6)
ELSE 0
```

Since 5-year age groups are frequently required for all ages 0 and up, i.e., 0 to 4 is the 1st group, 5 to 9 is the 2nd, etc., the standard codes for 5-year age groups are used here.

Note that since you need only groups 4, 5 and 6, you can use ELSE to indicate the category (0 here) into which all other ages will be put. That is, the age of a person is within the 15 to 29 of the RECODE or ELSE the age is recoded to 0.

Filtering a DEFINE with FOR

In the first output that we require, you are interested only in persons aged 5 through 29. The variable 90person.attstat that you RECODEd above, can be filtered to include only these persons by adding a FOR clause to the DEFINE command, thus:
DEFINE 90person.attstat
AS RECODE 90person.90attend
  (0  =0)
  (1  =1)
  (2-3 =2)
FOR 90person.90age >= 5 AND 90person.90age <=29

This carries out the RECODE considering only those records, here persons, for whom the FOR is True; when the expression is False, 90person.attstat is coded as NotApplicable, which is an internal code of winR+.

Naming with VARLABEL and VALUELABLES

Newly DEFINEd variables do not have descriptive labels unless they are explicitly specified; of course each new variable always must have its entity.variable name. The categories of new variables are born with numerical codes that are nameless. Giving descriptive labels is optional, but normally worth the effort, since it greatly facilitates understanding the CmdSet, and most important, the output tables. The use of the clauses VARLABEL and VALUELABLES for naming variables and their categories, respectively, are shown in the full CmdSet for this Tutorial in Figure 30.

For example, in the first DEFINE, which is command (02)—abbreviated Cmd02—in Figure 30, the variable 90person.90attstat is given its name via the clause VARLABEL “School attendance status”; the name, with 40 or less characters, must be enclosed in quotes.

The syntax of the VALUELABLES clause in the DEFINE can be seen in Cmd02 and Cmd03, namely, each value followed by its label in quotation marks. While the VALUELABLES can be written as a continuous sausage (see the DEFINE of 90person.90age5Rec, Cmd03), it is much more difficult to read than when carefully structured as in Cmd02. The latter form is strongly recommended.
Note that none of, some of, or all of the codes of a new variable can be given VALUELABLES.
Figure 30  CmdSet for Tutorial 3b showing RECODE with the DEFINE command. Each of the main commands, i.e., RUNDEF, DEFINE and TABLE, is identified with an asterisk, number and separator line, e.g., *03 ------------------, to facilitate discussion in the text. The asterisk tells winR+ that it is a “comment” rather than a command line. NOTE: The equivalent R+ command file is given on page 211.

Stating the RANGE and TYPE of a new variable

RANGE:  This the lowest and highest values of the new variable. The RANGE of a new variable should always be included with the DEFINE command, and must be used if the variable is employed in a crosstabulation, frequency or average. This allows winR+ to work
faster and use less space since it knows beforehand the size of the output table. Cmd02 and Cmd03 show how the RANGE is written.

**TYPE:** Whenever possible, most variables should be **TYPE** INTEGER, since winR+ processes fastest with whole numbers without decimals. Later examples will show numbers which have decimal values and are **TYPE REAL**.

*It is important to note that the variables created with DEFINE exist only for the duration of the Run. Variables are normally not stored permanently in the Workspace, the winR+database or anywhere else unless you explicitly SAVE the variable — see Chapter 7 on Additional Topics and Procedures. For this reason, it is strongly recommended that you save your Command Sets, particularly when they have lengthy or complex DEFINEs that you might have difficulty re-writing. This not only protects you against loss, but allows you to CUT and PASTE when you need the same or similar variables.*

**Using the new variables in TABLES**

Cmd04 in Figure 30 is the **AVERAGE** of 90person.90age by educational type and the variable that you created, 90person.attstat. You might want to use the **General Assist**, but the new variable is not on its entity and variable lists (it will be provided in later versions of winR+). If you do not want to rewrite the complete variable name, risking a possible spelling error, you can, as in other Windows programs, **Copy and Paste** it from the **DEFINE** Cmd02. Simply highlight the entity.variable and then **Copy** it by pressing **CTRL-C**. Move the cursor to where you want to **Paste** and press **CTRL-V**. Alternatively you can **Copy and Paste** using the Edit|Copy and Edit|Paste on the main menu.

Note that the Guided **DEFINE** and **TABLE Assists** do show newly created variables while you are in the Command Editor of the CmdSet with the new variables. Hence, the **TABLE Assist** could be used to create **Cmd04**.

In the output table of **Cmd04** — shown partially in Figure 31 — , the number of **Not Applicable** cases, 25,396, is given at the bottom after the table, itself. These are the cases that winR+ ignored, in this case due to the **FOR** of the **DEFINE** in Cmd02. Note that the "Not Applicable" category **inside** the table is a dictionary category for this
variable and was assigned by the census authorities to exclude persons not at any educational institution; it is not a result of using a FOR in a \textit{winR+} command.

<table>
<thead>
<tr>
<th>Profes. Institute</th>
<th>9</th>
<th>20</th>
<th>23</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
<td>57</td>
<td>69</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Not Applicable</th>
<th>10</th>
<th>6</th>
<th>14</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1871</td>
<td>641</td>
<td>2712</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total</th>
<th>11</th>
<th>21</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13820</td>
<td>12973</td>
<td>26793</td>
</tr>
</tbody>
</table>

\textbf{Figure 31.} Partial view of the output of the Average in Cmd04 of the CmdSet in Figure 30. The figure below the table, 25,396, is the number of \textit{Not Applicable} cases that were excluded by the FOR of Cmd02.

In the Crosstabs \textbf{TABLE} in Cmd05 (Figure 30), the FOR includes only persons between 15 and 29 years old and accepts only persons who attend or attended a specialized school (codes 4 through 9). The cases eliminated by the \textbf{TABLE...FOR} are not stated on the table.

Note that the crosstabulation output \textbf{TABLE} has three variables; the third variable is presented as the “panel” or control variable. That is, for each value, here \textit{male} and \textit{female}, of the third variable, there is a row \textbf{BY} column crosstabulation of the first two variables (see \textbf{Figure 32}).

\textbf{Defining a new variable at a higher entity level [Tutorial 3c]}

Most of the examples in this manual have involved a single entity, usually the person level. However, it is often necessary to \textbf{DEFINE} new variables and obtain results that involve more than one level, e.g., determining the average number of persons in a household, or determining the dependency ratio in a city block. As noted previously, \textit{winR+} has quite powerful capabilities for carrying out such hierarchical processing without your having to worry about how it is being done — of course, you do have to know what you want and how to interpret the results —.
Variable #3: 90PERSON.90SEX - Male (1)

<table>
<thead>
<tr>
<th>Categories</th>
<th>90AGESREC</th>
<th>90ATTEND</th>
<th>Attending</th>
<th>Attended</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-19</td>
<td>4</td>
<td>50</td>
<td>33</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>20-24</td>
<td>5</td>
<td>8</td>
<td>79</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>25-29</td>
<td>6</td>
<td>-</td>
<td>62</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>56</td>
<td>174</td>
<td>232</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variable #3: 90PERSON.90SEX - Female (2)

<table>
<thead>
<tr>
<th>Categories</th>
<th>90AGESREC</th>
<th>90ATTEND</th>
<th>Attending</th>
<th>Attended</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-19</td>
<td>4</td>
<td>107</td>
<td>36</td>
<td>143</td>
<td></td>
</tr>
<tr>
<td>20-24</td>
<td>5</td>
<td>31</td>
<td>89</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>25-29</td>
<td>6</td>
<td>1</td>
<td>36</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>139</td>
<td>161</td>
<td>300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 32. Output for a crosstabulation of 3 variables (Tutorial 3b). The third variable, 90sex, is the “panel” variable.

As an example of how to DEFINE variables for a higher entity level based on lower-level variables, we shall create a cross-tabulation of: “Household Size BY the Sex of the Household Head” for Puerto Nuevo. Neither of the variables is in the Nueva Miranda database, but the data to create them are available. One way to obtain the tabulation —there is often more than one way to obtain the outputs that you require— is through the CmdSet shown in Figure 33, which is then explained command by command. The results are given in Figure 34.
Figure 33. CmdSet for Tutorial 3c with an example of hierarchical processing.

Comments on the CmdSet in Figure 33

**Cmd 02: Counting the persons in the household.** It is necessary to **DEFINE** the total number of persons in the house, that is, you want to count the number of elements of the entity **90person** which are in each element of the entity **90housin** — and, of course, within the Selection Area **Tutor2a** —. In the *winR* language, “how” a **DEFINE** is done is always indicated in the AS clause. In this case, AS with **COUNT** tells the software to count the members of the entity **90person**. *winR* knows that it is counting **within** the entity **90housin**, since the entity level of the new variable is given by its full identification, the entity **variable** that is stated in the **DEFINE** line.

**Determining the RANGE of a variable with CODEBOOK.** You do not know the **RANGE** of this variable, but your could make a quick **FREQUENCY** to obtain the information. However, *winR* provides a
simpler means of determining the minimum and maximum values of new variables through a Codebook facility. When you do not know the range, add the keyword CODEBOOK after the unknown RANGE keyword of the variable that you are defining, so that the command reads, in the present case:

```
DEFINE 90housin.totpers
   AS COUNT 90person
   TYPE INTEGER
   RANGE CODEBOOK
```

Now RUN the CmdSet. You will receive a report with hypertext links to the Codebook information for the new variable.

<table>
<thead>
<tr>
<th>Codebook of 90BLOCK.TOTPERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total records checked :</td>
</tr>
<tr>
<td>Records out of range :</td>
</tr>
<tr>
<td>Not Applicable :</td>
</tr>
<tr>
<td>Minimum encountered :</td>
</tr>
<tr>
<td>Maximum encountered :</td>
</tr>
</tbody>
</table>

If you do not have a specific RANGE indicated, you will get the CODEBOOK information, but you cannot request a TABLE with the variable in question since winR+ will indicate that you have not provided the RANGE. One solution to this is to write the RANGE clause with an estimated minimum and maximum value followed by the CODEBOOK keyword. On a second run, if you underestimated the range, you can use the actual minimum and maximum values and remove CODEBOOK.

The codebook range for a given DEFINE takes into account a FOR expression, if any. That is, the value of the variable is included in the range only when the FOR expression is True.

In the CmdSet shown in Figure 33, a few values have been given VALUETABLES to help you and others “read” the output table.

**Cmd 03: Sex as a household variable.** You now want to define a variable that contains the sex of the household head, which is called here
Chapter 5. A walk through winR+: [Tutorials] — 91

HHsex. Since this variable describes the household and not a person, it is at the household entity level, 90housin. Thus, the command starts with DEFINE 90housin.relatsex, telling winR+ the information it needs to start the construction of the variable.

You want to look at each person in the household to see whether the person is the head (code 1) and if so, take the appropriate action, that is, FOR 90person.90relat = 1, DEFINE 90housin.HHsex AS 90person.90sex. That is, the person’s sex get assigned to the household sex variable only when the FOR expression is True.

You also have to take into account two other possibilities: there may be unoccupied houses —some towns of New Miranda have many summer homes that were unoccupied at the time of the census— and there may be occupied households that do not have a household head —this should not occur in the data, but there may have been errors in the census control procedures—. If you want to examine these possibilities, you can assign a default value to HHsex that will be used in these cases. Here, 0 is used via OPTIONS DEFAULT 0. If this is not used, these cases are treated as NotApplicable and will not be included in the cross-tabulation.

You can make better use of the hierarchical processing capabilities of winR+, if you have a basic understanding of how it works. From the point of view of the user, winR+ may be conceived as reading and acting on each entity within a SelSet in the order of the hierarchically structured winR+ database, starting from the lowest entity and working upwards. Within a given entity, each element is processed in the order that it is found in the database. Thus, in a SelSet for the 1990 census branch of the New Miranda database and a CmdSet working with persons and households, the first person (entity 90person) in the first household is read and acted upon, then the second person, etc., until all persons in the first household are processed. Then the variables for the first household, itself (entity 90housin), are read and acted upon. Next the persons of the second household enter and are acted upon one by one until the information on the second household is read in, and so on, until all the households are processed. The process is continuous and, it is important to
recognize that once a given element of an entity passes through, it cannot be recalled for post-processing. Hierarchical processing will be further discussed in a Chapter 7 (see page 156).

Crosstabs of variables at different entity levels [Tutorial 3d]

Another aspect of the winR+ ability to manage hierarchical processing for the user involves the production of CROSSTABS with variables at different entity levels. For example, as an exercise, run the CmdSet in Figure 35.
The results are shown in Figure 36. The row variables refers to a condition of the household, namely the existence of drinking water in the house, while the column variables applies to the persons in the house. In such cases, winR+ bases the tabulation on the elements of the lower entity, i.e., \textit{90person}. In this example, there are 924 males, and 853 females, respectively, with water in their households.

\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Categories} & \textbf{Male} & \textbf{Female} & \textbf{Missing} \\
\hline
no response & 0 & 421 & 275 & - \\
within household & 1 & 924 & 853 & - \\
outside household & 2 & 487 & 433 & - \\
Total & 1832 & 1561 & - & 3393 \\
\hline
\end{tabular}

Figure 36. Output for Tutorial 3d: CROSSTABS with variables at different entity levels, \textit{90Housin} and \textit{90person}. 
\{siC2E3d.wpg\}.
Saving a TABLE as a spreadsheet for post-\textit{winR+} manipulation

Although \textit{winR+} has a number of features for changing the appearance of frequencies, crosstabulations and averages, you may often want to perform more complex calculations on the output data provided by \textit{winR+} and/or to make a more sophisticated layout of a table for final publication than is possible in \textit{winR+}. These manipulations are normally most easily done in one of the major spreadsheet packages.

You can use the following procedure to save any \textit{winR+} frequency, crosstabulation, or average in a spreadsheet format. As an example, you can use the output from the CmdSet for Tutorial 3d (see Figure 36 on page 93).

\textbf{Procedure}

1. Display the output TABLE, in this case that of Figure 36.

2. Open the \textit{Save Output as...} window by clicking on File|Save Output as... on the main menu.

   - \textbf{Select the file Type}, using the combo box at the bottom left of the \textit{Save Output as...} window. The .\textit{xls} format for Microsoft \textit{Excel} is available — the other file types shown, .\textit{txt} and .\textit{dat}, give unformatted ASCII files in the first release of \textit{winR+} —. The default directory shown in the window is your work directory; you can select any other.

   - Give the file a name and click the OK button.

3. View in a spreadsheet package that \textbf{reads .\textit{xls} files}. If you now start, for example, Microsoft \textit{Excel} — since you are in Windows you do not have to close \textit{winR+} —, you can \textbf{OPEN} the output file to make whatever changes you wish.
TUTORIAL-4: The winR+ language for AREALISTs

Up to this point, you have learned how to obtain frequencies and cross-tabulations for a given area, i.e., your SelSet. But what if you want to identify the blocks in a city with “high” percentages of young children, or want to display a poverty indicator on a map to see where there are clusters of adjacent poor blocks in the city, or want to target a sales plan based on concentrations of homes with certain housing and occupant characteristics. For any of these, you could use a TABLE command in conjunction with AREABREAK to provide information on sub-areas within a SelSet, but the output will not be in a convenient form for these and similar problems, which require the use of lists of geographical areas and their characteristics.

The winR+ software provides a simple solution to the efficient production of lists by area through a special type of crosstabulation that is obtained by using the TABLE command with an AREALIST clause; this gives an output in which the rows are a listing of areas, normally with their codes, and the columns are variables. In any given AREALIST output, the areas in the rows of the list are the elements at a given entity level within the database’s geographical hierarchy, e.g., in New Miranda, the elements could be individual counties, districts or city blocks, etc., at the county, district, 90block, etc entity levels, respectively. Naturally, as always in winR+, the SelSet defined by the user determines the set of specific areas that are included in any given AREALIST output.

Figure 39 on page 103 shows an AREALIST at the 90block entity level; each row is a block and the column at the far left gives the codes for each block. Another AREALIST, where the rows are counties with names, may be seen in Figure 55 on page 148.
This Tutorial will show you how to obtain AREALIST outputs and show you:

- More on aggregated variables by counting with COUNT.
- How to create an AREALIST with aggregated variables
- How to use the basic features of the AREALIST Browser
- How to include percentage distributions in an AREALIST.

Aggregated variables for an AREALIST [Tutorial 4a]

More on counting elements in each of many areas

When making an AREALIST, such as of city blocks or neighborhood units, it is often necessary to determine the number of persons or houses, etc., with specific characteristics that are in each of the areas. Since this aggregate information, per se, is normally not in the database, as you know from Tutorial 3c (page 87), the winR+ language has the clause COUNT for ascertaining these variables.

In the first exercise of this tutorial, for the city blocks within the SelSet, you are asked to obtain frequencies for the following variables aggregated from the 90person level to the 90block level:

- 90block.totpers: Total number of persons in each block
- 90block.totfem: Total number of females on each block
- 90block.totmales: Total number of males on each block

and a variable aggregated from the 90housin to the 90 block level:

- 90block.tothous: Total number of houses on each block

Also you are asked to define an average for use in the next exercise:

- 90block.AvPerHous: Average number of persons per house on a block

The Command Set for this exercise is shown in Figure 37.
Chapter 5. A walk through winR+: [Tutorials]  — 97

<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(01)</em></td>
<td>RUNDEF T4a_Quan</td>
</tr>
<tr>
<td></td>
<td>RUNTITLE &quot;RunTitle: Tutorial 4a: Quantify&quot;</td>
</tr>
<tr>
<td></td>
<td>SELECTION tutor2b</td>
</tr>
<tr>
<td><em>(02)</em></td>
<td>DEFINE 90block.totpers AS COUNT 90person</td>
</tr>
<tr>
<td></td>
<td>VARLABEL &quot;Tot pers on block&quot;</td>
</tr>
<tr>
<td></td>
<td>TYPE INTEGER</td>
</tr>
<tr>
<td></td>
<td>RANGE 0-200</td>
</tr>
<tr>
<td><em>(03)</em></td>
<td>DEFINE 90block.totfem AS COUNT 90person</td>
</tr>
<tr>
<td></td>
<td>FOR 90person.90sex = 2</td>
</tr>
<tr>
<td></td>
<td>VARLABEL &quot;Tot females per block&quot;</td>
</tr>
<tr>
<td></td>
<td>TYPE INTEGER</td>
</tr>
<tr>
<td></td>
<td>RANGE 0-100</td>
</tr>
<tr>
<td><em>(04)</em></td>
<td>DEFINE 90block.totmales AS COUNT 90person</td>
</tr>
<tr>
<td></td>
<td>FOR 90person.90sex = 1</td>
</tr>
<tr>
<td></td>
<td>VARLABEL &quot;Tot males per block&quot;</td>
</tr>
<tr>
<td></td>
<td>TYPE INTEGER</td>
</tr>
<tr>
<td></td>
<td>RANGE 0-100</td>
</tr>
<tr>
<td><em>(05)</em></td>
<td>DEFINE 90block.totHous AS COUNT 90housln</td>
</tr>
<tr>
<td></td>
<td>VARLABEL &quot;Tot Houses on block&quot;</td>
</tr>
<tr>
<td></td>
<td>TYPE INTEGER</td>
</tr>
<tr>
<td></td>
<td>RANGE 0-100</td>
</tr>
<tr>
<td><em>(06)</em></td>
<td>DEFINE 90block.AvPerpHou AS 90block.totpers/90block.totHous</td>
</tr>
<tr>
<td></td>
<td>FOR 90block.totHous &lt;&gt; 0</td>
</tr>
<tr>
<td></td>
<td>VARLABEL &quot;Average Pers/house for blocks&quot;</td>
</tr>
<tr>
<td></td>
<td>TYPE REAL</td>
</tr>
<tr>
<td></td>
<td>OPTIONS</td>
</tr>
<tr>
<td></td>
<td>DEFAULT 99</td>
</tr>
<tr>
<td><em>(07)</em></td>
<td>TABLE TableID_Tut4a_QuanFreq AS FREQUENCY</td>
</tr>
<tr>
<td></td>
<td>OF 90block.totpers, 90block.totmales, 90block.totfem, 90block.totHous</td>
</tr>
<tr>
<td></td>
<td>OPTIONS</td>
</tr>
<tr>
<td></td>
<td>TITLE &quot;TableTitle: Freq&quot;</td>
</tr>
</tbody>
</table>

Figure 37  Command Set for Tutorial 4a: Using COUNT to count the elements (persons) in an area (blocks) to obtain aggregate variables at the block entity level.

Note that the ranges of each of the variables has been estimated relatively high. Since the range is used by winR+ to determine the size of the space to reserve when making the table, overestimation of the ranges are not serious for a FREQUENCY. However, if the variables are to be used in Crosstabs, actual ranges should be used since the size of the space reserved is the product of the ranges (thus, for a 3 way Crosstab with variables of range 0-99, a total of 1,000,000 cells would have to be reserved). When Crosstabs are involved with large variables, a FREQUENCY or a RANGE CODEBOOK should be run first to determine the actual ranges.
Comments on the Cmset in Figure 37

**Cmd 03: Total number of females on each block.** Persons are counted, but only those for whom the **FOR** condition is **True**, that is, when the person is a female the 90person.90sex = 2 expression is **True** and the person is counted. Any expression that can be evaluated as either **True** or **False** for each case concerned can be used.

**Cmd 06: Average of two aggregate variables.** For each block in the SelSet, the total number of persons on the block is divided by the total number of houses on the block. Since the division may not result in an integer, the **TYPE** is **REAL**. This command also show how to guard against division by zero, since there might be a block without any houses. Using **OPTIONS**, the **DEFAULT** is set to a value that is unlikely to be the result of the calculation; here the **DEFAULT** value is taken as 99 and will be used by **winR+** whenever the **FOR** condition is **False**. Although not relevant here, it should be noted that the **DEFAULT** can also be an arithmetic expression involving one or more variables.

Note that the name of the variable **AvPerpHou** contains upper and lower case letters. This is ignored by **winR+**, but makes reading the Cmset a little easier.

**Cmd 07: FREQUENCY for INTEGER variables.** A **FREQUENCY** is requested for each of the **TYPE** **INTEGER** aggregate variables. It cannot be requested for **REAL** variables.

*A simple trick can be used to obtain the **FREQUENCY** of a **REAL** variable. If the variable, say **var_real** has a single digit plus a decimal and you want to know the frequency to 2 decimals, you can **DEFINE** a new variable named, say **var_realtmp** which is **var_real** multiplied by 100 and use **TYPE** **INTEGER** with a **RANGE** 0-1000 (or **RANGE** 0-10000 if the original variable had two digits to the left of the decimal). A **FREQUENCY** of **var_realtmp** provides the information desired, of course, mentally replacing the decimals in the output.*

*A REAL variable can also be **RECODEd** to a set of categories, which can then be tabulated as an **INTEGER** variable. The classification facilities of **winR+** can assist in systematically assigning categories to **REAL** variables in an **AREALIST**.*
Creating an AREALIST [Tutorial 4b]

Aggregate variables at the AREALIST entity level

Figure 38 gives the Command Set used to create the AREALIST with the aggregate variables generated in the previous exercise. This CmdSet was written by making various changes and additions in the CmdSet of Tutorial 4a. After making the changes File|Save As was used to save the rewritten CmdSet in the Workspace with a new name (and label).

As the procedure for making an AREALIST begins with understanding how to write the CmdSet (Figure 38), we shall first comment on the commands and then outline the procedure for obtaining and viewing the AREALIST.

Comments on the CmdSet in Figure 38

The equivalent command file (.spc) written in the R+ language is given on page 212.

Cmd 07: AREALIST with all variables at the same entity level.
The OF clause of the AREALIST must begin with the name of the entity that defines which areas (the entity elements) of the SelSet will be listed, here the individual blocks of 90block. After defining the entity level, the variables to be included in the listing are given, separated by commas. If a variable is REAL, the format must be given as $x.y$ after the variable, with a blank space between the name and the format. $x =$ number of digits before the decimal and $y =$ the number of digits after the decimal (thus, 89.784 has the format 2.3).

It is very convenient that all the variables in Cmd 07 are aggregated to the 90block level since, in effect, this means that each has a single value for a given block as seen in the resulting AREALIST in Figure 39. However, if we had included a variable like 90person.90sex from a lower entity level, there would be a problem, since it has as many values as persons in the block and what to include for a given area is ambiguous. The next exercise will show how winR+ deals with this to provide useful information in an AREALIST.

Cmd 07: Saving the OUTPUTFILE to the Workspace: You must specify the type and name of the OUTPUTFILE for your AREALIST. Since AREALISTs are likely to be used in further operations with winR+, they often will be kept in your Workspace. In the present case,
after the keyword OUTPUTFILE, the filetype WORKSPACE is used followed by the name, here taken to be: T4b_ALA, but any name of 8 letters or less can be used without a period or an extension.

<table>
<thead>
<tr>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>*01 RUNDEF T4b_ALA</td>
</tr>
<tr>
<td>*02 DEFINE 90block.totpers</td>
</tr>
<tr>
<td>*03 DEFINE 90block.totfem</td>
</tr>
<tr>
<td>*04 DEFINE 90block.totmales</td>
</tr>
<tr>
<td>*05 DEFINE 90block.totHous</td>
</tr>
<tr>
<td>*06 DEFINE 90block.AvPerpHou</td>
</tr>
<tr>
<td>*07 TABLE T4b_ALA</td>
</tr>
</tbody>
</table>

Figure 38 Command Set for Tutorial 4b: An AREALIST at the block level with aggregate variables at the same level. Note: The equivalent command file for R+ is given on page 212.

Procedure

1. Write, SAVE and RUN the CmdSet. See Figure 38. As with other TABLEs, an output report will appear with hypertext (see page 54) that will give you a list of the variables, the number of records, etc.
Chapter 5. A walk through winR+: [Tutorials] — 101

2. View the AREALIST in the Workspace. In general, you can access any AREALIST and any time by selecting "Arealist" in the Document Type combo box of the Workspace and opening the one with the name that you used in your CmdSet, here "T4bALA". If the another type of OUTPUTFILE was used, you would go to the directory where you placed it.

When creating an AREALIST, it is usually easier to view it by:

Clicking on the hypertext for the AREALIST in the Output Report of the Command Editor that automatically appears when the processing is completed; an output summary will appear followed by another window containing the AREALIST Browser in which you can view and manipulate the AREALIST.

Looking at the AREALIST columns of Figure 39 — each column of which is a variable (also often called a field) — you will notice that the results appear consistent since for each area the total number of females plus males equals the number of TotPers.

3. Print the list. If you want, you can print the list in the order selected. Before printing, you may wish to make adjustments in the page layout including headers and footers, by using File|Page setup (see the figure on 71 and the associated text). The font is changed in Tools|Preferences and then clicking on the Printing Tab.

When you are ready to print, make sure that the AREALIST is the active window and then use File|Print. You can print only the first page, or a few pages, by indicating the range of pages in the Print window that appears, e.g., From 1 To 1, to print only the first page.

Becoming familiar with the AREALIST Browser

As you know, the columns of an AREALIST matrix, such as that seen in Figure 39, are each a variable; sometimes the variables are called fields. The rows of the AREALIST are the individual records of the areas. The code of each area, which are elements of an entity — in this case, of the 90Block entity — are given in the far left column.
The records of a given AREALIST always are the elements of a single entity level. For example, they can be blocks or districts, but not a mixture of blocks and districts. Furthermore, in the present release of winR+, the records are always elements of a selectable entity; thus in New Miranda, the records never refer to persons or houses, since these are non-selectable.

You can scroll down the records of the list and use the Top and Bottom buttons, соответственно, to reach the beginning and end of the list rapidly. Similar buttons with triangles alone alongside the Top and Bottom buttons, can be clicked to move the line cursor Up or Down the records—a triangle in the left-most column moves up or down—. You can also use the window scroll bar(s). If they are not visible, the scroll bar(s) will appear after clicking in the body of the table; of course, if the list is short and fits within the window, the scroll bars will not be shown.

Selecting an area (record): Within the Browser, you can also move the line cursor to an area by clicking anywhere in its record, i.e., row. You can select an area by clicking on the left-most column, the record becomes highlighted, i.e., selected. Deselect an area by clicking again in the left-most column. The number of areas selected is indicated at the bottom of the AREALIST Browser window. This window also provides buttons to move Up and Down from one selected area to the next selected area (ignoring the non-selected areas) and to the Top and Bottom of the selected areas: соответственно.

Sorting the list. You can sort by any of the variables on the list by selecting the variable of interest from the combo box at the top of the window: соответственно. The order of the sort is determined by the variable or field (column) of the AREALIST that you select in the combo box; natural order is according to the geographical hierarchy in the database, i.e., by the codes of the entity, in this case 90block. The order can be made Ascending or Descending, respectively, by using the first or second button at the right of the combo box.

These and other aspects of the AREALIST Browser, including how it can be used to manipulate records and columns, add new columns, make
complex selections of rows and interface with other systems such as GIS are discussed in a section of Chapter 7, starting on page 162; if you frequently work with AREALISTS, it is strongly suggested that you peruse these pages.

Compared with R+

If you are familiar with the more sophisticated aspects of R+, you will recognize that the AREALIST is, in effect, like WRITE command in the R+ language. In the WRITE, a record is written for each observation at the lowest entity level of the variables in the command; the AREALIST is more explicit since it requires the user to specify the entity level of interest. A major difference between the WRITE of R+ and the AREALIST of winR+ is that the former could be written for unselectable entities such as persons, while the latter can only be written for selectable entities.

Figure 39. AREALIST for Tutorial 4b with aggregate variables, all at the 90Block entity level.

Variables below the AREALIST entity level: Frequencies

As noted in the previous section, variables at an entity lower than that defined for the AREALIST do not have a single value for each area. In this situation, winR+ inserts the numerical distribution of the lower-entity variable at the AREALIST entity level, using a column for each category of the variable.

For example, 90person.90sex in an AREALIST at the district level will include two columns for this variable: Male=1 and Female=2, and for
each area will give the number of males and females, respectively. In effect, then, there is a frequency distribution of the variable sex for each area.

To see how winR+ works in this case, add the variable 90person.90sex to the variable list after the OF in Cmd 07 of the CmdSet T4b_ALA so that the command reads:

```plaintext
*07 ---------------------------------------
TABLE t4b_AreaList_AggregVar90Block
  AS AREALIST
  OF 90block,
      90block.totpers, 90block.totfem, 90block.totmales
      90block.totHous, 90block.AvPerpHou 2.2,
      90person.90sex sex
  OUTPUTFILE WORKSPACE T4b_ALAD
  OPTIONS
      TITLE "AreaList:Aggreg/Distrib var for 90Block"
```

For clarity here, the changes are shown in italics and a larger font. Do not forget to add the comma after the previous variable or you will get an error. Now SAVE AS the entire CmdSet with the name T4b_ALAD or any other name you wish and that distinguishes it from the previous CmdSet.

**Giving a variable an alias.** Note that an alias, sex, has been added after the variable 90person.90sex in the OF list for the AREALIST. The alias is used as the column heading for the variable in the output and can be used with any variable on the list. However, there are three mandatory situations when a suitable alias must be used:

a. When the short varlabel name, 90sex in this case, starts with a number.

b. When the short varlabel name is identical with another on the list; at least one of these must be given an alias to differentiate it from the other.

c. When a short varlabel name has more than 9 characters; the alias must have 9 or less characters. If the variable is at a lower entity level, it or, its alias, should not have more than 7 characters to allow
Chapter 5. A walk through winR+: [Tutorials] — 105

for the digits that will be used to indicate the category of the variable (if the range is between 0-9, an 8 character name can be used).

After RUNning the CmdSet, you should find an AREALIST in your Workspace similar to that shown in Figure 40.

Note that, in the first release of winR+, if you re-Run the same CmdSet with the same AREALIST name to be saved in the Workspace, winR+ will give a Compile error since it finds that the AREALIST already exists. Either erase it in the Workspace or change the name in the CmdSet for the next Run.

Since the variable 90person.90sex is at a lower entity level than the areas of the AREALIST 90block, the number of males (sex1) and females (sex2) are given in each block.

Of course, the frequency distribution of the persons by sex in each block is the same as the TotMales and TotFem aggregated variables obtained via COUNT. In such situations, it is recommended that you use the Frequency capacity of AREALIST and do not DEFINE new variables.

![AREALIST](siALdT4b.wng)

Figure 40. AREALIST for Tutorial 4b with aggregate variables at the AREALIST entity level (90block) and the variable 90person.90sex below it. The latter is given as a frequency of its codes in each area. (siALdT4b.wng)
You can put variables in an AREALIST at a higher entity level than the level defined for the AREALIST. Thus, in the present example, the list is defined at the 90block entity, so a district level variable could be used. But the higher entity variables cannot be ones that are calculated during the process, such as by using COUNT or SUM, since the AREALIST is constructed, say, at the block level, as the processing of each block is completed. Thus, a COUNT at the district level would not be finished when all but the last block was entered into the AREALIST. Hence, only existing dictionary variables can be used at a higher entity level in an AREALIST.

DBF and other filetype alternatives for an AREALIST

The AREALIST outputs created in the above Tutorial exercises were placed in the Workspace via the OUTPUTFILE WORKSPACE keywords. However, you may wish the outputs to be placed in a file in an external directory. winR+ provides the following filetype alternatives for the OUTPUTFILE of an AREALIST:

<table>
<thead>
<tr>
<th>File type</th>
<th>File specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKSPACE</td>
<td>Workspace document</td>
</tr>
<tr>
<td>TEXT</td>
<td>.txt extension</td>
</tr>
<tr>
<td>ASCII</td>
<td>.dat extension</td>
</tr>
<tr>
<td>DBF</td>
<td>dBASE III .dbf and .ndx</td>
</tr>
<tr>
<td>MDB</td>
<td>Microsoft Access</td>
</tr>
</tbody>
</table>

The Workspace document will, of course, be created in the current Workspace. All the other files will be placed, by default, in the current “work” directory that you defined in the File|Properties of the Main Menu of Windows for the winR+ item or that you indicated in your Preferences. If you want the output file to be placed in a specific directory other than the work directory, you must give the full path with the filename (the .dbf extension of the file is optional), e.g.,

OUTPUTFILE DBF c:\scratch\4b_ALA.dbf
As when creating an AREALIST to be stored in your Workspace, winR+ will give a Compile error if you try to create an external AREALIST that already exists. However, unlike the case of the Workspace, the first release of winR+ provides OPTIONS OVERWRITE to automatically replace an existing AREALIST .dbf file with another of the same name. Use OVERWRITE with care, since the system will not ask whether you want to replace the existing file —it just does it—.

Viewing the AREALIST .dbf file. When you have just created the file, you can use the same procedure to view an external AREALIST file as used on page 101 to display a Workspace document, i.e., click on the Output Report hypertext entry for the AREALIST. However, once the Command Editor is closed, winR+ in the first release cannot view this external .dbf file, or any other, without first importing into the Workspace as an AREALIST.

Exporting an AREALIST to a spreadsheet package

You may wish to export an AREALIST to a spreadsheet (or other package), for example, to manipulate the fields, dress up the list for publication, or do complex graphics. In the first release of winR+ this can be done most easily by creating the AREALIST as a .dbf file, or exporting a Workspace AREALIST as a .dbf.

Exporting from the Workspace can be done by clicking on File|Export and then using the Export as... window to create and save a .dbf, normally using as the default the dBASE III format, which can be read by Excel and other spreadsheet packages. For a more sophisticated transfer of an AREALIST to a target system, see page 196.
Chapter 6. Command sets: Further learning by examples

COMMAND SETS: FURTHER LEARNING THRU EXAMPLES

If you have followed the Tutorials of the two previous chapters, you now should have enough knowledge to use most of the beginner and intermediate level winR+ features. It is clear that one of the major keys to winR+ is the intelligent use of the command language. To provide you with more experience with CmdSets, introduce a number of additional command keywords and give you some tips and tricks, this chapter will present and discuss selected CmdSets. These illustrate specific aspects of the language that you may wish to "copy" or adapt for your own purposes.
Each CmdSet example is preceded by an outline of the problem to be solved and the steps that are used to obtain a solution (the algorithm). As noted previously, there often will be other ways of obtaining the same results. Then, after showing the actual CmdSet, we shall comment on specific commands that were used. Generally — except for some of the topics that will be treated in the following chapter — it will be assumed that you already know how to use the various winR+ windows, menus, Assists, etc., to create, run and view the CmdSets.

The following are the examples included in this chapter:

<table>
<thead>
<tr>
<th>Example</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Five-year age groups by a formula</td>
<td>110</td>
</tr>
<tr>
<td>Number of generations in a household</td>
<td>113</td>
</tr>
<tr>
<td>Average age of persons in a household</td>
<td>117</td>
</tr>
<tr>
<td>The age of the oldest person in a household</td>
<td>119</td>
</tr>
<tr>
<td>The sex of the oldest-aged person in a household</td>
<td>121</td>
</tr>
<tr>
<td>The age of the youngest person in a household</td>
<td>123</td>
</tr>
<tr>
<td>Household overcrowding</td>
<td>124</td>
</tr>
<tr>
<td>WEIGHTing TABLEs</td>
<td>126</td>
</tr>
<tr>
<td>Combining variables to create a new variable</td>
<td>128</td>
</tr>
<tr>
<td>The number of items in a household</td>
<td>130</td>
</tr>
<tr>
<td>Identifying areas with sequences of adjacent houses with the same characteristics</td>
<td>133</td>
</tr>
</tbody>
</table>

**Five-year age groups by a formula**

The age of a person is an essential variable in most socio-economic analyses. Demographic studies usually group the single-year completed ages that are reported in a census or survey, into 5-year age groups, normally 0-4, 5-9, 10-14, ..., 95-99. The completed age of a person refers to the age at the last birthday on the day of the census or survey, e.g., both a person aged 5 years and 1 day and a person aged 5 years and 11 months will each report an age of 5 years. Since census and survey data is usually coded with single years of age in a database (to permit, for example,
studies of single years of age or to permit other groupings), the analyst must create the 5-year groupings when required.

You have already done this by a RECODE (see page 82), but this is rather lengthy if all 20 5-year age groups are required. A much simpler approach is to use a formula which involves integer division in which the result is always an integer and, therefore, TYPE INTEGER; normal division gives a TYPE REAL decimal result. The CmdSet to create the 20 5-year age groups is shown in Figure 41.

Comments on the CmdSet in Figure 41

The frequencies obtained in Cmd 04 allow you to compare the result from the integer division calculation in Cmd 02 with the result using a RECODE in Cmd 03 (only the first few categories are recoded to save space). A frequency is also included for single years of age to facilitate checking by summing the ages.

Cmd 02: Calculation of 5 year age groups via integer division.
The expression on the second line of the command, i.e.,

\[
\text{AS (90person.90age)\backslash5 + 1}
\]

calculates the integer codes because the division operator is the \text{backslash} \text{"\backslash"}, which indicates to \text{winR}+ that the division must result in an integer; any decimal, i.e., fractional remainder, is ignored. For the first few ages, the results of this command are:

<table>
<thead>
<tr>
<th>Age</th>
<th>Age\div5 REAL value</th>
<th>Truncated value</th>
<th>+1 Final code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
In effect, the backslash division uses completed values and ignores the remainders, just like the census measurement of age! Thus, use of the backslash gives the results desired, e.g., for age 99 ($99 \div 5 = 19 + 1$), so the code is 20). The inclusion of the +1 is optional; it is used here to avoid a 0 category.

"01__________________________
RUNDEF Age5grp
  RUNTITLE "Five year age groups by calculation"
  SELECTION tutor2a
"02__________________________
DEFINE 90person.90age5C
  AS (90person.90age)\(5 + 1\)
  TYPE INTEGER
  RANGE 1 - 20
  VARLABEL "5 yr agegrps calculated"
"03__________________________
DEFINE 90person.90age5Rec
  AS RECODE 90person.90age
    (0 - 4 =1)
    (5 -9 =2)
(10-14 =3)
(15-19 =4)
(20-24 =5)
(25-29 =6)
(30-HIGHEST = 7)
TYPE INTEGER
RANGE 1-7
VARLABEL "5 yr agegrps by recode"
"04__________________________
TABLE age5a
  AS FREQUENCY
  OF 90person.90age, 90person.90age5C,
  90person.90age5Rec
  FOR 90person.90age5C <=6

Figure 41. Command Set to obtain 5-year age groups by calculation and check of results.

Related topics

**winR+ operators**

The following are valid operators for winR+:

**Arithmetic operators**

+  Addition
-  Subtraction
*  Multiplication
/  Normal division. **TYPE REAL** with decimals
\  Integer division. **TYPE INTEGER**, truncated to lower integer
^  Exponentiation. E.g., $5^2 = 25$

**MOD** Modulo $y$, gives remainder after dividing by $y$. E.g., $5\text{MOD}2 = 1$
Relational operators
= Equal to
>= Greater than or equal to
<= Less than or equal to
> Greater than
< Less than

Logical operators
AND And
OR Or
NOT Not

Number of generations in a household

In a study of family structure, you might want to know how many generations with respect to the Household Head (HHH) there are in a household. That is, based on the codes used to define the relationship to the HHH in the New Miranda database, the HHH is the base generation, the HHH’s parents are a generation higher, the HHH’s children are a generation lower, and the HHH’s grandchildren are two generations lower. The variable 90housinngen, number of generations in a Household, can be created by counting how many of these generations are present in a given household. A CmdSet to count the generations in a household is shown in Figure 42.

Comments on the CmdSet in Figure 42

Cmd 02 defines the HHH generation. The existence —yes or no— of the HHH generation is established by defining a household level variable 90housin.HHHHead to have the value of 1 when a HHH is found in the household and zero when there is none. Since it is at the household level, the variable is reset to the DEFAULT value, zero, just before the first person of the next household is processed.

Cmds 03 to 05 identify the existence of parents, children and grandchildren of the HHH, i.e., 90relat = 7, 4 and 6, respectively. The value of each of the three DEFINEd household variables, 90housin.parent, .child and .gndchild is the DEFAULT value of 0 for a given household, until at least one person with the specified code is processed. It changes to 1, say, when a person within the household is processed and is identified as a child, i.e., the FOR expression is evaluated as True. The value of the variable “sticks” at 1 whether
additional children are processed in the house or not; additional children just repeat the value of the variable AS 1. These are yes/no variables indicating that the generation exists (value 1) or does not exist (value 0) in the household.

```
*01__________________________
RUNDEF HHGener
  SELECTION tutor2a
  RUNTITLE "Generations with respect to HH Head"
  *02__________________________
  DEFINE 90housin.HHHead
  AS 1
  FOR 90person.90relat = 1
  TYPE INTEGER
  RANGE 0-1
  OPTIONS DEFAULT 0
  *03__________________________
  DEFINE 90housin.parent
  AS 1
  FOR 90person.90relat = 7
  TYPE INTEGER
  RANGE 0-1
  OPTIONS DEFAULT 0
  *04__________________________
  DEFINE 90housin.child
  AS 1
  FOR 90person.90relat = 4
  TYPE INTEGER
  RANGE 0-1
  OPTIONS DEFAULT 0

*05__________________________
DEFINE 90housin.gndchild
  AS 1
  FOR 90person.90relat = 6
  TYPE INTEGER
  RANGE 0-1
  OPTIONS DEFAULT 0
  *06__________________________
  DEFINE 90housin.ngen
  AS 90housin.parent + 90housin.HHHead + 90housin.child + 90housin.gndchild
  TYPE INTEGER
  RANGE 0-4

*07__________________________
TABLE 1
  AS FREQUENCY
  OF 90housin.ngen
  OPTIONS
    TITLE "NGen respect to HHHead, with/without HHH"
  *08__________________________
TABLE 2
  AS FREQUENCY
  OF 90housin.ngen
  FOR 90housin.HHHead = 1
  OPTIONS
    TITLE "NGen respect to HHHead, when HHH present"
```

Figure 42. Command Set for determining the number of generations in households.

**Cmd 06 adds the yes/no generation variables.** The new 90housin.ngen should vary from 1 to 4, since the census collection and coding rules of New Miranda state, as in most countries, that every household has a Head, so that there is at least one generation present. In fact, we have taken the range as 0-4, just in case there is a household without a Head (see “Related topic”, below).
Chapter 6. Command sets: Further learning by examples — 115

Cmd07 gives the distribution of the number of generations. Note that this TABLE assumes that there is, in fact, a HHH in each household. As you will see in the next section, this is not a good assumption in New Miranda.

Related topics

DEFAULT values vs. NOT APPLICABLE values

Various of the DEFINE commands in this CmdSet have DEFAULT values which are used when the FOR expression is evaluated as False. Yet in other situations, the OPTION DEFAULT has not been used. Is there a difference?

There is an important difference. When no DEFAULT value is given and the FOR expression is False, winR+ assumes that the variable is Not Applicable to the element being processed and assigns its internal code for this. This value is outside the RANGE and cannot be used in further operations. When a TABLE is made with a DEFINEd variable that has Not Applicable values, the Not Applicable cases are excluded from the output table.

On the other hand, when you explicitly state a DEFAULT value, the variable takes on that value when the FOR expression is False. In this case, the RANGE will normally include the value, as in the yes/no variables in this CmdSet, and the value can be used in further operations, such as the addition of the variables in Cmd06.

LIKE for writing less

In the present CmdSet, the generated variables are similar to each other, and the author has had to repeat the same RANGE, TYPE and OPTIONS for each. This can be particularly onerous when there are lists of VALUELABELS. In such cases, effort and space can be saved by using the keyword LIKE <varname> in a DEFINE command, where the <varname> is any variable that has been DEFINEd previously in the CmdSet or which appears in the dictionary. For example, in the present CmdSet, Cmd 03 could be written:

DEFINE 90housin.parent
   AS 1
   FOR 90person.90relat = 7
   LIKE 90housin.HHHead
where LIKE tells winR+ that the variable .parent inherits the properties of the earlier variable .HHHead, except for properties explicitly stated for the new variable. Thus, if the VALUELABELS for .parent were given as 0 “No HHHead” and 1 “HHHead”, a different set of VALUELABELS could be used for .parent, in which case, only the TYPE, RANGE and OPTIONS will be inherited.

Do not forget that variables in your present CmdSet can inherit the characteristics of variables that appear in the dictionary using LIKE. The variable used as the model does not have to be at the same entity level.

Hierarchical processing limitation: Move up, Stay up

Now that you know how to obtain the number of generations in a household, you may want to use it. Unfortunately, in the present release of winR+, there are logical limitations on how a derived variable can be used, remembering that it must be used in the same run as when it is defined.

Since winR+ processes all the persons before processing the household of the persons, and all the persons and their houses before the city block, and so on up the geographic hierarchy, the new 90housin entity variable 90housin.ngen cannot be used in a CROSSTABS that involves both it and lower entity 90person entity variables, say, to study the effect of multiple generations in a household on the fertility of the individual women in the household. This is because, by the time the .ngen has been calculated for a household, the persons of the household are gone. The only means of dealing with this is to add the new variable to the winR+ database using SAVE (this is discussed in Chapter 7; see page 153) and then process a another CmdSet that uses the now-available variable. But in any given run, if you create a new variable moving up the hierarchy, your output with the variable can involve only other variables at the same entity level as that of the newly created variable.

Compared with R+

In Redatam-Plus, this could be done via the relatively complex GENERATE process.

Care of bugs in your base

The data collected in a census normally are subjected by the National Statistical Office to an editing process to detect logical errors, e.g., a female of 5 years cannot have borne children, or in this case, a household with persons in it must have a Head by definition. Unfortunately, it is easy to see in the latter example, that this is not quite true in New
Miranda. **Cmd 08** remakes the frequency of **Cmd 07** but places the **FOR** condition on the **TABLE** that the HHH is present. If you run the two frequencies as in the example, you will note that there are three households in the Selection Set that have persons but no household head—an error in the quality control of the New Miranda Statistical Office.

### Average age of persons in a household

To calculate the average age of the persons in a household (or a block or any higher entity), you must add the ages of all the persons in the entity of interest. One way to do this is to directly “add together” or “sum” the ages before dividing by the total number of persons in the entity. This is done in **winR+** by using **SUM** as seen in the CmdSet shown in **Figure 43**.

**Comments on the CmdSet in Figure 43**

**Cmd02** uses **SUM** to add the ages of persons in the household. The new variable, **90housin.sumHHage**, is **DEFINE**d at the entity level to which the completed **SUM** will apply, i.e., **90housin**. It is the **SUM** of a lower entity variable, in this case, **90person.90age**. Note that no parenthesis are used with the **SUM**. The **RANGE** could be taken as 0-1000, which includes 0 since there may be empty houses and 1000 is likely to include all possible cases (see the Related Topics section below, for establishing the actual **RANGE**, 0-423 for the Selection Set used).

**Cmd03** to obtain the number of persons in the household. This uses a **COUNT**. Note that, contrary to good CmdSet writing, no **RANGE** has been given. **winR+** is forgiving here since this variable is not used in a **TABLE** during the run.

**Cmd04** calculates the average age. Note that the **TYPE REAL** is used since the average in any given case is unlikely to be an integer. Note also that care has been used to avoid dividing by zero (which will occur when a given house is unoccupied). When the **FOR** is **False**, i.e., the divisor is zero, the **OPTION DEFAULT** value of zero is used.

**Cmd05** recodes the **REAL** variable to an **INTEGER** to permit a **FREQUENCY** in **Cmd06**. A **FREQUENCY** is necessary to examine the distribution of the average age. Since a **TYPE REAL** variable cannot be used in a **FREQUENCY** or **CROSSTABS**, it can be **RECODE**d to an **INTEGER** variable.
Figure 43. Command Set to determine the average age of persons in a household.

Related topics

Use of integer division

Another approach that will give single years of age and allows a FREQUENCY is to use Integer Division (see page 111) on the average calculated in Cmd04; that is, using backslash division by 1 of the TYPE REAL average age, 90housin.avageHH. Since the answer is always the truncated value, you will get “completed” average age. The command would look like:
The age of the oldest person in a household

The hierarchical processing capabilities of winR+ can be used to determine the age of the oldest person in each of the households of your Selection Set. The result, after processing all the person records within a household, will be a variable for the oldest age at the household entity level, which in the CmdSet in Figure 44 is called: 90housin.oldest.

Compared with R+
The equivalent command file (.spc) written in the R+ language is given on page 213.

Comments on the CmdSet in Figure 44

Cmd02 identifies the oldest age in the household. The records of the persons in a given household are processed sequentially. The OPTIONS DEFAULT 0 fixes the initial oldest age, 90housin.oldest, as zero. Then when the first person of the household is processed (generally the Head of Household, but this is not a requirement of the algorithm), the FOR expression is evaluated and normally the person's age, 90person.90age will be greater than 0. Since the FOR is True, the AS is carried out and the first person's age is assigned to 90housin.oldest. When the next person of the household is processed, his/her age will be tested by the FOR to see if it is older than the previous person. If so, 90housin.oldest takes on the new "oldest" age. If not, 90housin.oldest keeps the value it has, since up to this point, that is the oldest age. This continues until all persons in the household are processed.
Figure 44. Command Set to determine the oldest age in each household. *NOTE: The equivalent command file for R+ is given on page 213.*

**Cmd03 does a CROSSTABS of the oldest age by a characteristic of the household,** in this case, whether it has a refrigerator or not.

### Related topics

**The age but not the person**

Note that you have the oldest person’s age, but not who that person is. It is not possible in the present release of winR+ to mark the oldest person and then, in the same run, to go back to obtain the characteristics of the marked person because winR+ makes a single pass through the data (special exceptions will be discussed on page 156 and a general solution on page 153). All the records of the household were processed to find the oldest age, which may well have been the first of various persons in the household. Since hierarchical processing winR+ cannot go backwards or recyle through the persons a second time now knowing which age to look for, we seem to be limited to an analysis of the oldest age itself and not its owner. The example in the next section, gives a solution that will apply in many instances.
The sex of the oldest-aged person in a household

In the previous example, we found the age of the oldest person in the household and did a tabulation using the age. But, as indicated in Related topics for that example, you may want to examine the characteristics of the person and this presents certain difficulties because winR+ makes one pass through the data. However, as you can see in the CmdSet shown in Figure 45, there is a way to solve the problem. In this case, we wish to tabulate the age of the oldest person by the sex of the oldest person.

Compared with R+

The equivalent command file (.spc) written in the R+ language is given on page 214.

```
*01
RUNDEF oldestSx
  SELECTION tutor2a

*02
DEFINE 90housin.oldestAg AS 90person.90age
  FOR 90housin.oldestAg<90person.90age
  VARLABEL "Age of oldest pers in HH"
  OPTIONS
    DEFAULT 0
    TYPE INTEGER
    RANGE 0-99

*03
DEFINE 90housin.oldestSx AS 90person.90sex
  FOR 90housin.oldestAg=90person.90age
  VARLABEL "Sex of oldest pers in HH"
  LIKE 90person.90sex
  TYPE INTEGER
  RANGE 1-2
  OPTIONS
    DEFAULT 0

*04
TABLE oldestSx
  AS CROSSTABS
  OF 90housin.oldestAg by 90housin.oldestSx
```

Figure 45. Command Set to determine the sex of the oldest age in each household. NOTE: The equivalent command file for R+ is given on page 214.

Comments on the CmdSet in Figure 45

Cmd02 locates the oldest person in the household; see the comments on this command on in the previous CmdSet on page 120.
Cmd03 saves the sex of the oldest person found so far in the household, just in case it turns out to be the oldest person when all the persons of a household have been processed. That is, when a person enters to be processed in Cmd02, if the person is the oldest up to then, his/her age is assigned to 90housin.oldestAg. The processing of the same person continues in Cmd03, where the FOR expression tests to see whether the person’s age is that presently held as the oldest in variable 90housin.oldestAg; if so, the sex of that person is saved in 90housin.oldestSx. If the person being processed is not the oldest as determined in Cmd02, then Cmd03 does not save the sex of the person and the value of 90housin.oldestSx remains that of an earlier person from the household that was processed and found to be the oldest up to then and continuing to the present person.

When a given household is completed for all the commands in the CmdSet, a new household begins and 90housin.oldestAg is reset to the DEFAULT value of 0 and the whole process begins again.

Of course, if two persons have the same age and are the oldest of the household, the sex finally kept will be that of the second person to be processed.

In Cmd03 the RANGE of 90housin.oldestSx is taken as 1-2. Note that the actual RANGE is 0-2, since empty houses will take the DEFAULT value of 0 because there is no oldest person. This eliminates the 90housin.oldestSx = 0 column for empty houses which otherwise would have appeared in the Cmd04 with CROSSTABS of oldest age by oldest sex.

Note that LIKE 90person.90sex was used to give the VALUELABELS to the 90housin entity variable defined in this command.
Chapter 6. Command sets: Further learning by examples

The age of the youngest person in a household

Essentially the same algorithm as that used to find the age of the oldest person in a household can be used to find the age of the youngest. The CmdSet is shown in Figure 46.

```
*01
RUNDEF youngest
SELECTION tutor2a

*02
DEFINE 90housin.youngest
AS 90person.90age
FOR 90housin.youngest > 90person.90age

OPTIONS

DEFAULT 99
TYPE INTEGER
RANGE 0-99

*03
TABLE youngest
AS CROSSTABS
OF 90housin.youngest by
90housin.90refrig

```

Figure 46. Command Set to find the youngest age in a household.

Comments on the Command Set in Figure 46

Cmd02 finds the youngest age; the logic in Cmd02 is the same as that in previous example. In the present case, the OPTION DEFAULT value of the variable is set to 99 (assuming that is the highest age recorded in the census). The first person is processed, and if younger than 99, the variable 90housin.youngest is set to the first person’s age. Then the second person is compared to the first and if younger, the age is replaced, and so on until all the persons in the household are processed.
Household overcrowding

In this example we would like to determine households that are overcrowded. For purposes here, a household is overcrowded if it has more than 3 persons on average per room designated for sleeping. That is, if the

\[ \text{Totpers} / \text{Number.sleeping.rms} > 3 \text{ or the equivalent} \]

\[ \text{Totpers} > 3 \times \text{Number.sleeping.rms} \]

then the household is classified as “overcrowded”. This will illustrate the idea, although a more complex measure could be devised, for example, taking into account age.

This example makes use of AS with a logical expression that winR+ evaluates only as being either True or False, which are represented by 1 and 0, respectively. See the Related topics section of this example for more information. The CmdSet is given in Figure 47.

Compared with R+

The equivalent command file (.spc) written in the R+ language is given on page 215.

Comments on the CmdSet in Figure 47

Cmd02 calculates the total number of persons in each household, as has been done in various Tutorials and other examples.

Cmd03 defines 90housin.ovcrowing AS 1 or 0, depending on whether the expression is True or False, respectively. The expression is the inequality defining overcrowding that we gave above. Thus, the overcrowding variable is 1 when the household is overcrowded and its RANGE is 0-1.
The use of AS with a logical expression is very convenient, but must be used with caution. The result must be unique for any element of the entity of interest, in this case households of the entity 90housin. That is, there must be only one possible result. If the variables in the expression refer to the same entity level, as here, there is no problem, but if the expression were (90housin.totpers > 3 AND 90person.90sex = 2), the result will vary as different persons of the household are processed and the final evaluation of the expression for the household will be accidental depending on the sex of the last person listed in the household.

An expression to DEFINE a household as having a female Head of Household, AS (90person.90relat = 1 AND 90person.90sex = 2) will give a unique result in all cases, if the census has been correctly conducted so that there is one and only one Head in each household—a assumption that should be tested in the data before proceeding.

Cmd04 is a CROSSTABS of overcrowding according to Household size. The FOR clause eliminates the cases when the house is unoccupied or the total number of rooms for sleeping in the house is zero, which presumably is an error in the census, i.e., 90housin.totpers and 90housin.90soroom must each be greater than 0.
Related topics

**Boolean variables**

The result of a Boolean expression such as used with AS in this CmdSet is a *Boolean variable*. The evaluation of the expression is *True* and *False*, represented by a 1 and a 0, respectively. The variable that is **DE**FIN**E**d in this way is taken as **TYPE** **INTEGER**. Boolean, *per se*, is not supported in *winR*+.

**Concern for data quality**

Although this example gives a working algorithm for determining over-crowding, the New Miranda data may not be acceptable for this use, since a rather large number of occupied houses have zero rooms. As noted previously, *winR*+ will give you results, but the interpretation depends on the user and the appreciation of the quality of the data utilized.

**WEIGHTing TABLEs**

You may require a variable like the number of *children ever born alive* (CEB) for each 5-year age group of females in the reproductive ages (normally assumed to be 15-49 years of age). However, if you make a normal **CROSSTABS** of CEB by 5-year ages groups, such as done in **Cmd06** of Figure 48, you will not get directly what you want, since a **CROSSTABS** gives the number of cases for each value of the CEB tabulated. For example, **Cmd06**, for a given age group, gives you the number of persons who have 1 child, the number who have 2 children, etc. —we shall ignore an error in the New Miranda data that seems to have mixed women with zero CEB with persons for whom the data are missing or not applicable and who may have been coded with a blank—. If you want the total number of CEB to a given age group, you have to multiply the number of children in the 1, 2, 3, etc., categories by their respective numbers of woman. Much easier is to use the keyword **WEIGHT** `<expression>` in the **TABLE** of interest. Its use is explained in the comments on the CmdSet shown in Figure 48.
Chapter 6. Command sets: Further learning by examples — 127

Comments on the CmdSet in Figure 48.

RUNDEF Cmd01 has a UNIVERSE clause to limit all results to women in the reproductive ages, 15-49.

Cmd02 creates 5-year age groups by calculation using the backwards slash for integer division as was discussed on page 112.

Cmd03 and 04, non-weighted and weighted FREQUENCYs, respectively, which are "identical" except for the WEIGHT factor in the latter. In Cmd04, each female is "WEIGHTed" by the number of CEB she has had, using the clause:

\[
\text{WEIGHT} \text{90person.90nchild}
\]

The WEIGHT here is a simple variable, but it could be an expression involving one or more variables and constants.

Note that a FOR has been used in these and all other TABLEs in this example, since the DICTIONARY indicates that codes 98 and 99 are for the non-applicable and out-of-range values of 90person.90nchild; of course, these codes are not legitimate weights and the women in these categories have to be excluded.

Cmd05 is an AVERAGE of CEB by age group to provide the information that shows that the AVERAGE for a particular age group multiplied by the number of women in the age group give the same results as in the WEIGHTed Cmd04.

Cmd06 shows that a normal CROSSTABS gives the number of women in each CEB and age category and not the cumulative CEB in each age group.
**Figure 48.** Command Set to obtain the Total number of Children Ever Born (CEB) by 5-year age groups of women in the reproductive ages by using a weighting factor equal to the CEB for each woman.

### Combining variables to create a new variable

There are many instances when you have to combine codes of two (or more) variables to make a new variable, for example, marital status and the sex of the person. Of course, there must be no ambiguity of the category into which a given person falls. A simple algorithm for unique sums can be used.

The following table shows the logic graphically. Assuming one digit variables such as Marital Status and Sex, the value of the first is multiplied by 10 and the second by 1. As can be seen from the table, any
Chapter 6. Command sets: Further learning by examples — 129

combination of categories gives a unique sum, e.g., a single person who is female has the value of \((3 \times 10) + (2 \times 1)\) or 32. If VALUELABELS are given to each of the final codes, it is usually not necessary to RECODE, although the 10 codes below could be recoded to 1 to 10.

<table>
<thead>
<tr>
<th>Marital Status</th>
<th>Code</th>
<th>x10</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>Married</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Consensual</td>
<td>2</td>
<td>20</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>Single</td>
<td>3</td>
<td>30</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>Widow/er</td>
<td>4</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>Separated</td>
<td>5</td>
<td>50</td>
<td>51</td>
<td>52</td>
</tr>
</tbody>
</table>

The CmdSet to carry out this algorithm is shown in Figure 49. Normally, it would be part of a larger CmdSet.

```
*01.-----------------
RUNDEF Combivar
SELECT tutor2a

*02.-----------------
DEFINE 90person.MarSt_Sex
AS (90person.90MarSta*10) +
90person.90sex
FOR 90person.90MarSta <> 0
TYPE INTEGER
RANGE 11-52
VARIABLES "Marital status and sex"

*03.-----------------
VALUELABELS 11 "Married-M"
12 "Married-F"
21 "Consen-M"
22 "Consen-F"
31 "Single-M"
32 "Single-F"
41 "Widow-M"
42 "Widow-F"
51 "Separ-M"
52 "Separ-F"

TABLE F1
AS FREQUENCY
OF 90person.MarSt_Sex
```

Figure 49. Command Set showing how to create a new variable combining from the values of two variables (each of which as one digit).
Comments on the Command Set in Figure 49

Cmd02 carries out the unique-sums algorithm to create the new variable desired, that is,

\[ \text{New output variable} = (10 \times \text{input variable#1}) + (1 \times \text{input variable#2}) \]

or in this specific case:

\[ 90\text{person.MarSt_Sex} = (10 \times 90\text{person.90MarSt}) + (1 \times 90\text{person.90sex}) \]

The \text{FOR} eliminates persons who have "No Response" = 0 codes for Marital Status; Sex is not included in the \text{FOR} since all persons have an identified sex. \text{VALUERELABS} have been assigned to facilitate reading the output.

The number of items in a household

Frequently, it is necessary to determine how many items are possessed on a list. The items might be characteristics of a household such as whether there is potable water, a good roof, etc. One way to do this, is to convert each item first to a \text{Yes/No} variable with the values 1 or \text{Yes} and 0 for \text{No}. Then simply adding the variables will give the number of items with \text{Yes}. Such a procedure, often to create poverty indicators using a Basic Needs approach, is shown in Figure 50.

Comments on the Command Set in Figure 50

\text{Cmd02 to Cmd05 each convert an item into a Yes/No variable.} Each uses the Boolean variable result when \text{winR+} evaluates an expression as \text{True or False} (see page 124), returning the values 1 and 0, respectively, where 1 means that \text{Yes} the item exists and 0 means that \text{no} it does not exist. Note that it is assumed that the item does not exist in the household if there is a non-response. Note also that after fully writing out the information for the first item (\text{Cmd02}) we have used \text{LIKE} for the other items, changing only the information that is different (see page 115).

\text{Cmd06 determines the number of items possessed} by adding up the individual items, each of which has value of 0 or 1. The minimum is
Figure 50. Command Set to determine the number of items possessed on a list after converting each item to a Yes/No variable.

to have none, and the maximum, here 4, is to have all.

**Cmd07 is a FREQUENCY of the number of items.** There are a very large number of households with no items, suggesting that many houses are unoccupied and should be eliminated from the table. After obtaining the number of persons in each household in Cmd08, Cmd09 produces a FREQUENCY including only houses with one or more persons present.
Sequential numbering of elements in an entity

You may occasionally need to number the elements within an entity, such as the blocks within a district, the persons within a household, etc. Once you have these numbers, you have a unique way to identify specific elements. For example, in the CmdSet in which we identified the oldest person in a house and his/her sex (see the CmdSet on page 120), we could not make various tabulations concerning the socio-economic situations of such persons. But if each person had a number, say 90person.number, a variable could be created, e.g., 90housin.numoldest, that indicated the number of the person who was oldest in the house. This could be SAVED to the winR+ database (see page 153), and the TABLEs made with the filter:

FOR 90housin.numoldest = 90person.number

A simple means to number the elements within a higher entity is illustrated in Figure 51. In this case, we have sequentially numbered the blocks within a district, since the results can be seen in an AREALIST — the same approach can be used for numbering the persons in a house but the results cannot be seen since persons are non-selectable —.

Comments on the CmdSet in Figure 51

Cmd02 counts the number of blocks in each district, although this is not our real interest. Rather as each block is processed, the counter, of the COUNT goes up by 1, so the counter is at 1 for the first block, 2 for the 2nd block etc.; the counter is held in variable district.numblock. When a number district begins, the counter is reset to 0 and the count starts again as this district’s blocks are processed.

Cmd03 DEFINES the block number by taking the counter value of district.numblock for the block being processed and saving it in 90block.BlkNum. Thus, each block receives a number.

Cmd04 makes an AREALIST in which the sequential number can be seen in the column BlkNum (not shown).
## Identifying areas with sequences of adjacent houses with the same characteristics

What if a planner wants to identify blocks in a city with groups of contiguous housing units with a condition in common, such as a low level of education of the household head, or the presence of a female household head, or the lack of basic necessities, etc. If this could be done with existing census data, it would provide the information far more rapidly and at much lower cost than conducting a new survey. Making the reasonable assumption that the census enumerators normally have carried out their interviews following the order of the houses on the block (moving clockwise or counterclockwise from a starting point), it also can be assumed that the records of the houses enter in the same order. Based on this, an algorithm can be devised to determine when there are sequences of houses with the characteristic of interest.

In the CmdSet shown in Figure 52, the objective is to identify the blocks of the 90block entity that have sequences of at least 5 “contiguous” houses that meet a specific condition. In this example, a house meets the condition, i.e., it “IsQualified” for inclusion in a sequence, if it has more
than 3 persons living in it. A block “Isqualified” and is given a code to identify it as such when it has a least one sequence of 5 or more qualifying houses. Finally, an AREALIST is produced for **90block** entity.

```
*01-----------------------------------------------------
RUNDEF Sequence
SELECTION tutor2a
RUNTITLE "Blocks with sequence(s) of 5+
houses with totpers>3"

*02-----------------------------------------------------
DEFINE 90housin.totpers
AS COUNT 90person
VARIABLE "Total persons in HH"
TYPE INTEGER
RANGE 0-20

*03-----------------------------------------------------
*To qualify, HH must have > 3 pers
DEFINE 90housin.IsQualified
AS (90housin.totpers > 3)
TYPE INTEGER
RANGE 0-1
OPTIONS DEFAULT 0
VARIABLE "HH has more than 3 pers"
VALUE LABELS 0 "0-9 pers" 1 "3+ pers"

*04-----------------------------------------------------
DEFINE 90block.sequence
AS (90block.sequence +
90housin.IsQualified) *
90housin.IsQualified
TYPE INTEGER
RANGE 0-100
OPTIONS DEFAULT 0

*05-----------------------------------------------------
DEFINE 90block.IsQualified
AS 90block.IsQualified +
90housin.IsQualified
FOR 90block.sequence = 5
TYPE INTEGER
RANGE 0-100
OPTIONS DEFAULT 0

*06-----------------------------------------------------
TABLE SEQUENCE
AS AREALIST
OF 90Block, 90block.IsQualified
FOR 90block.IsQualified > 0
OUTPUTFILE WORKSPACE seq3
```

**Figure 52.** Command Set to identify blocks with sequences of 5 houses or more with a given characteristic, in this case, the number of persons living in the house must be greater than 3 persons.

**Comments on the Command Set in Figure 52**

**Cmd02** defines the variable to be used to see whether a household qualifies for a sequence for the criterion defined. In this case, we want total household size, but any other variable(s) at the level of the household entity could be used for the criterion.

**Cmd03** determines whether a given household “IsQualified” to be in a Sequence because it has more than 3 persons. This uses the fact that a True or False result of the expression for a given household
is expressed as 1 or 0, respectively; hence, $90\text{housin}.\text{IsQualified} = 1$
when the household meets the criterion.

**Cmd04** determines the length of a sequence, including the
given household being processed, and automatically resets to
0 when a household enters which does not qualify. This is the
key “trick” of the entire algorithm. The length of the sequence,
$90\text{block.sequence}$ starts with the DEFAULT value of 0. If the first
household on a block is qualified, the parenthesis in the AS expression:

$$90\text{block.sequence} + 90\text{housin}.\text{IsQualified}$$

is equal to $0 + 1 = 1$.

This is also the case, if a sequence was broken and this is the first
IsQualified household of a possible new sequence. Since the parenthesis
is also multiplied by $90\text{housin}.\text{IsQualified} = 1$, the value of the entire AS
expression is 1. If the next household also IsQualified, then the
parenthesis is $1 + 1 = 2$, which is multiplied by 1, so $90\text{block.sequence} = 2$. And so on.

However, if a sequence has started and $90\text{block.sequence}$ has any value
greater than 0, say 3 for example, and a non-qualified household unit
enters for processing, the parenthesis will be $3 + 0 = 3$, but because it is
multiplied by $90\text{housin}.\text{IsQualified} = 0$, the value of $90\text{block.sequence}$
is back to 0.

**Cmd05** determines whether a sequence of IsQualified house-
holds for a given block has reached 5, and if so, changes the
value of $90\text{block.IsQualified}$ to 1. That is, the $90\text{block.IsQualified}$
variable is initially set by DEFAULT to 0. When the FOR expression is
True, then the AS expression becomes $0 + 1 = 1 = 90\text{block.IsQualified}$.
If the sequence is longer than 5, it makes no difference. If there is more
than one eligible sequence within the block being processed, the variable
$90\text{block.IsQualified}$ will be $1 + 1 = 2$, etc.

**Cmd06** produces the desired AREALIST of the blocks that
have one or more sequences, and includes the number of such
sequences. The result is saved in the Workspace as an AREALIST
called seq3. If you remove the FOR or put an asterisk before it, you will
get the AREALIST of all blocks in the SelSet. Examining this
AREALIST in the Workspace and sorting on $90\text{block.IsQualified}$, you
can quickly identify all the blocks which have 1 or more sequences.
Related topics

Creating a Selection Set from an AREALIST

Normally the purpose of identifying the city blocks such as in this example, is to create a new Selection Set, which then is analyzed in detail using the database and perhaps some external files. The winR+ software has a built-in capacity to convert an AREALIST (or a sub-set of the list which can be chosen using SORT and Selection or the QUERY facility in winR+) into a SelSet. The procedure for this is treated in detail in the next chapter on Additional Topics and Procedures (see page 177).

It is also possible to write a CmdSet to create a SelSet directly without passing through the stage of an AREALIST. Again this is treated in the next chapter in the section starting on page 177.
Chapter 7. ADDITIONAL TOPICS and PROCEDURES

This chapter treats subjects which are likely to be beyond the initial needs of many users, and for some topics, possibly also beyond their background knowledge. Various matters are included that are not touched upon in the previous chapters or are examined in greater depth and complexity. Towards the end of the present chapter, the topics also involve linking the output of winR+ with other software, such as Geographical Information Systems (GIS), to allow spatial and other analysis.

The topics and procedures included in this chapter are:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database conversion &amp; Workspace management</td>
<td>138</td>
</tr>
<tr>
<td>Using external database variables</td>
<td>143</td>
</tr>
<tr>
<td>Saving DEFINEd variables in a winR+ database</td>
<td>153</td>
</tr>
<tr>
<td>Extending hierarchical processing in winR+</td>
<td>156</td>
</tr>
<tr>
<td>Working in the AREALIST Browser</td>
<td>162</td>
</tr>
<tr>
<td>Deriving a Selection Set from an AREALIST</td>
<td>177</td>
</tr>
<tr>
<td>Using AREALISTs with the winR+ map display</td>
<td>183</td>
</tr>
<tr>
<td>Moving beyond winR+ via the winR+ Data Link</td>
<td>196</td>
</tr>
</tbody>
</table>
Database conversion & Workspace management

The general procedure for creating a winR+ database and an associated Workspace from an existing Redatam-Plus for DOS database will be outlined here. Since the first release of winR+ does not have a winR+ database creation facility, it is assumed initially that Redatam-Plus databases will be converted to winR+ (see page 10).

The new Workspace is placed in your work directory. This directory is initially defined outside of winR+ in Windows, but it can be changed inside winR+ by clicking on Tools|Preferences to bring up the window that allows you to state your preferences, among which is the work directory (see Customization, page 16).

Converting a Redatam-Plus database to winR+

As only the dictionary of a Redatam-Plus database is converted, while the data remain exactly as before, the conversion of an existing Redatam-Plus to a winR+ database with a Workspace is very fast.

The converted winR+ dictionary will be placed in your Workspace, which is a Microsoft Access database and has the file extension .mdb.

Procedure

1. Initiate the conversion of a Redatam-Plus for DOS database dictionary to a winR+ database dictionary in a new Workspace by clicking on the New icon or on File|New on the Main Menu. A multi-tab form appears (see Figure 53).
2. Select the Create from R+ TAB by clicking on the Create from Redatam-Plus Tab.

- **Identify the R+ Database Directory** by clicking the upper Browse button which gives you the Redatam+ Database Directory window to help you identify the directory with the existing Redatam-Plus database that will be used to create the new winR+ Workspace.

  This directory must have the .def file, which contains the definition of the database in the DOS Redatam-Plus; when such a database is found in a directory, the box on the right displays the database name. Click OK to return to the Figure 53 window.

- **Define the Workspace name and directory**: As shown in the Figure, the creation window then displays a default Workspace name and default Workspace directory on the second line of boxes. You can change the name, if you wish, and browse the disk to change the Workspace directory.
• Create the converted dictionary and new Workspace by clicking the Create Workspace button to obtain the Workspace window that will then appear on the screen.

Creating a Workspace for an existing winR+ database

If you already have a winR+ database and a corresponding Workspace, you can make an additional Workspace with a different name.

Procedure

1. Open the Create New Workspace window (see Figure 53) by clicking on File|New or the equivalent icon.

2. Create another Workspace by clicking on the Create from Existing Tab (the resulting window is not shown here).

• Open the winR+ database of interest. Click on the lower Browse button in the From Workspace section of the window (make sure it is the lower button).

• Select your existing Workspace and the click on OK.

• Indicate whether you want the existing documents. If you want your documents (Command Files, Selection Sets, etc) transferred to the new Workspace, there should be a cross in the Include documents box; if you want an empty Workspace, click to remove the cross in the box.

• Make the new Workspace by clicking on the Create Workspace button.

Importing Workspaces, databases, documents

You can use a copy of a winR+ Workspace created on another computer if you have the corresponding winR+ database on your computer. However, even if you have the required database, unless you have it in the same directory structure as was used for the database on the other computer, winR+ will not be able to run a CmdSet from the new Workspace. As winR+ does not know where to find the database variable files and other information on your computer, you will get a “compile error”. 
To understand why, click on the Dictionary icon and look at the information on the Storage / Variable Tab of the variable definition window that appears when you double click on any variable; you will see that the File Name has a path associated with the variable name. This information must be supplied after you introduce a Workspace from another computer — called here the “imported Workspace” — and winR+ provides a simple procedure for doing so.

Initialization of an imported Workspace and/or database

Procedure

We shall assume that you have received a Workspace from another computer — along with the corresponding database, if you do not already have it — and want to use it to create a new Workspace on your computer. As noted above, these will be called the imported Workspace and imported database in the following procedure for creating the new Workspace and initializing the Workspace:

1. Load the Imported Workspace .mdb file and winR+ database into directories on your computer. If you have more than one winR+ database, the Workspace(s) associated with them should be in a separate directory, normally your “work” directory. Note that the database and Workspace files should never be in the same directory as the winR+ program.

2. Close your current Workspace in winR+, if it is open, using File|Close Workspace on the Main Menu.

3. Open the Create new Workspace window by clicking on File|New.

4. Open the Create from existing window by clicking on the second Tab.
   
   - **Indicate the Workspace directory** where you want the new Workspace to reside (or use the upper Browse button).

   - **Write In the name of the new Workspace** that you want to import in the left box of the New Workspace section of the window.
Locate the imported Workspace by using the lower Browse button of the From Workspace section of the window. It must be an .mdb file.

Make your new Workspace by clicking on the Create Workspace button. The Workspace appears.

If the directory structure for the database is different on your computer than on that from which it came, then you must continue with the initialization next steps. —Note that if the new database has the exact same paths to its files on both computers, the new Workspace is ready for use and the next two steps should be ignored—.

5. Initialization 1: Provide database “pointer” information to the new Workspace by clicking on Database|Advanced|Consistency Checks|Database Structure on the Main Menu Bar. A Locate Pointer File window will open to allow you to find the winR+ pointer files (.ptr) for the relevant database. Double clicking on the directory will show the .ptr files in the directory. When they are found, click on the OK button.

6. Initialization 2: Provide datafile information to the new Workspace by clicking on Database|Advanced|Consistency checks|Datafile Access. Locate the directory with the .bin files for the database; normally this will be the same directory as in the previous step. When ready, click OK.

If all is correct, you can write and run any CmdSet with the new database; to try this quickly, try an Easy Frequency.

Import and Export of Command Sets

winR+ provides facilities for importing documents from another Workspace or from DOS files and for exporting individual CmdSets and other documents to DOS files. Note that as DOS files, CmdSets and SelSets have extensions .dft and .sel, respectively.

The procedure for importing CmdSets from another Workspace will be illustrated here; SelSets follow essentially the same procedure. The procedures exporting, etc., are similar or simpler.

Procedure for importing a CmdSet

1. Open the your Workspace if it is not already open.
Chapter 7. Additional Topics & Procedures  —  143

2. Indicate the Document Type of interest in the Workspace combo box.

3. Click on File|Import|From Other Workspace on the Main Menu.

4. Select the donor Workspace directory with .mdb files in the Open winR+ Workspace window that appears (Figure not shown). Click on the directory after finding it in the directory tree so that the .mdb files for that directory appear in the left-hand window. Click on the Workspace .mdb file for the Workspace of interest. Click OK.

5. Select the CmdSet(s) by clicking in the left-most column on the CmdSet(s) of interest. More than one may be selected by holding down the CONTROL key when making the selections; if the various selections are contiguous, select the first and then hold down SHIFT and select the last of the set.

6. Import the CmdSet(s) into your Workspace by clicking on the Import and then Exit buttons. The CmdSet(s) imported should appear in the receiver Workspace. If the documents do not appear at first, try switching document modes and then returning to CmdSets.

Exporting of CmdSets as .dft files follows a similar procedure using File|Export on the main menu.

Using external database variables

As has been noted earlier, winR+ is an open system. That is, not only can it process a winR+ hierarchical database, but it can also directly utilize variables (also called fields) that are stored in non-winR+ databases. The variables of interest may be in one or more of the varieties of dBASE (.dbf) or, in later releases of winR+, a Microsoft Access database (.mdb). Note that dBASE is sometimes called xBASE since there are many varieties.

There are two ways of working with external variables in winR+:

a) Use them directly in a Command Set. This is advantageous when only a few external variables are required and they will be used
in only one or two processes. If there are many variables and/or they will be used frequently, it will be more convenient to:

b) Include the definitions of the external variables in your Workspace dictionary. This is the more powerful approach since the variables can then be accessed in your CmdSets like any other variables in the dictionary. The actual .dbf files are not “imported” into the winR+ database, but the dictionary has all the information to find and use them.

Since the variables remain in the .dbf files and outside the winR+ database, you can update individual .dbf records, etc., as frequently as you wish in commercial dBASE-compatible software. winR+ databases, on the other hand, are not oriented to frequent or individual record updating. Of course, if you change any field definitions of the .dbf, then you will have to update these definitions in the Workspace.

The two approaches are described in the two sections that follow.

Compared with R+  

\textit{R+ could only work with variables from its multisectoral ("multidisciplinary") database, winR+ not only has that capability, but also can work in a multisectoral environment directly accessing different databases from different sources.}

External variables in a CmdSet: DEFINE...AS DATASET

In the first approach to using external variables, a DEFINE command with an appropriate clauses is used to bring an external variable directly into your CmdSet. In effect, a “new” variable is DEFINEd using a field of the external database. Just as each variable in winR+ is associated with an entity in the winR+ hierarchical database, the incoming external variable must also be associated with an entity by DEFINing an entity.variable name. For this, the external database must have records which correspond to the elements of a selectable or non-selectable entity of your winR+ database.

Normally, the external database should have a field, called the KEY-FIELD, which matches the codes of the entity declared for the external variable when it is named by DEFINE entityname.variablename; the records in the external database normally should correspond to the order in the winR+ database, although if an INDEX of the external database is specified, this need not be true. How all this is done will be clearer from the CmdSet in Figure 54.
Chapter 7. Additional Topics & Procedures — 145

An AREALIST can be used as a source of “external” variables. Note that the variables (columns) of an AREALIST are not in the winR+ database format and refer only to the territory covered by the SelSet used during its creation.

*If the external source does not correspond fully to the area of the SelSet, of course, only those elements of the entity that do correspond, will have values on the external variable.*

The CmdSet in Figure 54 creates an AREALIST for counties of New Miranda in the format of a .dbf file for possible use in a GIS or another system. The variables required are the number of male and female births, the total number of births in 1990 and the number of women in the fertile ages 15-49, also for each county. The number of women are available in the winR+ database, but the birth information from the vital statistics registration at the county level are kept by New Miranda in a dBASE III database, nacimien.dbf.
The \textit{nacimien.dbf} database has two of the three aggregate variables that we require for each county, namely the fields \texttt{NaSexo1} and \texttt{NaSexo2}, which are male and female births, respectively. The total number of births will be \texttt{DEFINE}d from these in the CmdSet. The external database also has a field, \texttt{comuna}, which has the codes that correspond to the \texttt{county} entity of New Miranda. An index could be created in an external dBASE software for this \texttt{dbf} file, but since the counties are in the same order as the \texttt{winR+} elements of the \texttt{county} entity, an index is not necessary.
Comments on the Command Set in Figure 54

**Cmd01** selects ALL the *winR*+ database since the *nacimien.dbf* database is for all counties. In fact, if the SELECTION SET of interest was a SelSet that had, say, only three counties, it could be used with this database. Conversely, the external database need have only the same entity elements that are in the SelSet.

**Cmd02** counts the number of fertile women in each county using a standard COUNT.

**Cmds 03 and 04** use DATASET to include external variables for male and female births, *NaSexo1* and *NaSexo2*, respectively, from the *nacimien.dbf*, which are DEFINEd as variables *county.MaleBths1* and *county.FemBths2*, respectively. Note that these are only temporary variables for the run, but could be SAVED (not done in this example) to become *winR*+ database variables (see page 153).

After the keyword DATASET, you must specify the type of database, where those supported by *winR*+ are:

- **DB3** = dBASE III
- **DB4** = dBASE IV
- **FP2** = FoxPro 2.0
- **FP25** = FoxPro 2.5
- **FP26** = FoxPro 2.6
- **MDB** = Microsoft Access
- **WORKSPACE**

In this case, the database is a **DB3** and the path and name of the database is given, along with the **FIELD** name of each variable in the external database. Since there is no index, **INDEX * is used. Finally, the KEYFIELD** comuna of the external database is given to connect its records with the corresponding records for counties in *winR*+.

Note that if neither a KEYFIELD nor an INDEX is specified, then it is assumed that the external database corresponds one to one with the SelSet and that the number and ordering of records is the same in both; if not, there will be a processing error.

A sufficiently large **RANGE** has been given to each variable, and **CODEBOOK** has been added to get the actual range for future use. If a value is outside the **RANGE** on a record of the **AREALIST**, it will appear as a blank on that record.

**Cmd05** uses the two external variables to calculate a third, namely, the total number of births in each county.
Cmd06 creates the output AREALIST (Figure 55) which is stored here as a .dbf in the work directory (the path is given, but need not be for the current work directory). The usual rules apply to the AREALIST. The name of each county has been included using the variable from the New Miranda winR+ database: county.@county. The variable county.totalseg ("total agricultural cattle segments"), also from the winR+ database, has been included.

<table>
<thead>
<tr>
<th>County</th>
<th>@county</th>
<th>Totalseg</th>
<th>Females</th>
<th>Malebths</th>
<th>Femalesbths</th>
<th>Totalbirths</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Santa Maria</td>
<td>240.</td>
<td>5765.</td>
<td>878.</td>
<td>764.</td>
<td>1662.</td>
</tr>
<tr>
<td>6</td>
<td>Sanizago</td>
<td>88.</td>
<td>1849.</td>
<td>346.</td>
<td>325.</td>
<td>671.</td>
</tr>
<tr>
<td>7</td>
<td>Bolivia</td>
<td>140.</td>
<td>3463.</td>
<td>637.</td>
<td>607.</td>
<td>1244.</td>
</tr>
<tr>
<td>8</td>
<td>Marbella</td>
<td>22.</td>
<td>920.</td>
<td>122.</td>
<td>123.</td>
<td>245.</td>
</tr>
<tr>
<td>9</td>
<td>Puerto Nuevo</td>
<td>36.</td>
<td>813.</td>
<td>124.</td>
<td>111.</td>
<td>235.</td>
</tr>
</tbody>
</table>

Figure 55. AREALIST resulting from the CmdSet in Figure 54. There are 5 records corresponding to the 5 counties of New Miranda.

The variables on an AREALIST which has been saved as a .dbf can be used in another CmdSet as external variables. Thus, it is not always necessary to SAVE area-related variables into the winR+ database even if they will be needed in later processes.

Adding external variables to your Workspace dictionary

The second, and more powerful approach, is to define the external variables once and for all in your Workspace dictionary — in the first release of winR+ this can only be done with dBASE files —. Once this is done, the external variables can be used directly in a CmdSet as any other variables in your winR+ Workspace dictionary.

As in the previous DEFINE...AS DATASET approach for working with external variables, the individual dBASE files that you use must each refer to a single entity level in your winR+ database. For example, for New Miranda, you could work with files that have information on marriages in each county, one file for each of the years 1987 through
Chapter 7. Additional Topics & Procedures

1992 and you might also use a file with aggregate data on the number of stores and other non-residential buildings on each block. These can be used since they refer to the county and block entities, respectively, in the New Miranda winR+ database.

It is important to reiterate that the external variables continue to reside in the external file(s); they do not become winR+ database .bin variables unless each is specifically added to the winR+ database with a DEFINE...SAVE command (page 153 for a brief discussion of the logic of the winR+ data and file structure). Unless the dBASE file is very large (say all the blocks of a country) and static (no updating is required), it normally is preferable to leave the variables in their original .dbf files to permit easy updating with whatever commercial or other software that is used to create and maintain the external database(s). Then when you use variables from these files, they will be up to date.

Figure 56. Step 1 for adding dBASE variables to your winR+ dictionary identifies the database name and path, and the entity (DicDb1.wpg)

Figure 57. In Step 3 you provide information on the variable name, a descriptive label, the range, category labels, if any, etc. (SiDicDb3.wpg)

To show how to employ this second approach for working with external variables, we shall produce the same AREALIST as used to illustrate the DEFINE...AS DATASET approach (see page 145), but this time the information for the two variables NaSexo1 and NaSexo2, male and female births in each county, respectively, from the nacimien.dbf database, will first be included in your Workspace dictionary so that they are available for use in a Cmd Set. Note that the variables of a dBASE file are usually called fields and that this convention is used here on the winR+ database.
screens; once included in the winR+ dictionary, they will be called variables, as usual.

**Procedure**

1. **Open the dictionary by clicking** on its icon, or using **Database|Dictionary** on the main menu bar.

2. **Open the Add variables from xBASE file window** (see Figure 56) by clicking on **Edit|Associate xBASE file** that appears on the Main Menu Bar when the Dictionary is open (xBASE and dBASE are used here interchangeably).

3. **Fill-in Step 1: Define input window** (Figure 56).
   - **Identify the xBASE file of interest.** After confirming that the **Input type** combo box indicates xBase (in the first release of winR+), click the **Open input file** button to choose the desired file, in this case, **nacimien.dbf**, in the window that opens. After pressing **OK**, its complete path appears in a few seconds in the xBase input box of the **Step 1: Define input** window. The entire path allows winR+ to know from where get the variable when you later use it. Note that the **xBASE field list** box (see **Figure 56**) is only to help you check that you have the correct database.
   - **Select the related entity in the winR+ database**, here, **county**, since the **nacimien.dbf** has a record for each county.
   - **Select the key field in the .dbf database**, which in this case must refer to counties. This could have the name **county**, but in the **nacimen.dbf** database, this field is **comuna**. You will find it at the very bottom of the combo box list.
   - **Press the Next >> button.**

4. **Fill-in Step 2: Select the variables window** (Figure not shown).
   - **Select the variables** to be included in the Dictionary by clicking on each one of interest in the **Input Field List** box to pass each of the selected variables to the [Dictionary] **Member Variable List** box. Click on a variable in the second box to deselect it. You can also Select All or Clear All from the respective lists.
• Press the Next button.

5. Fill-In Step 3: Complete Variable Definition window (see Figure 57). For each variable in the Input field combo box, which lists all that you selected in the previous screen:

• Give the variable a name. You can use the same name as the field, or invent a new name. In the present example, the fields NaSexo1 and NaSexo2 can be given the names, MaleBths and FemBths, respectively (names of 8 characters or less).

• [optional] Give the variable a descriptive label.

• Provide the Range (required) and Type (required). You may wish to look at the values of the variable in an external database browser. Here, the number of births in each county is taken to vary between 0 and 1000.

• (optional and if relevant) Provide the value labels for each code. In this case, the number of births is quantitative and the values do not need labels. However, if the variable were, for example, Existence of one or more hospitals in the county, the labels would be written in the form:

  1 “Yes, hospitals in the county”
  2 “No hospitals in the county”

Note that the labels are in quotation marks. You may have to look up the labels of the codes in the external database system.

• Press the Next button

6. Final step. Add the variables in the dictionary window
(Figure not shown).

• Send the information about the variables to your Workspace Dictionary. If you agree with the summary of the information that you entered, click the Save variables in dictionary button, and then Exit.

• If you need to make corrections, press the Previous button to return to the prior screen to make changes.
Using the new variables in your Workspace dictionary

If you now look at the variables for the entity *county* in your dictionary—you may have to refresh the screen by clicking on another entity and then on *county*—you will see that the two new variables have appear on the variable list. These variables can now be used like any other dictionary variables.

The CmdSet in Figure 54 (page 146) employed the DEFINE...AS DATASET approach in Cmd03 and Cmd04 to use these same dBASE fields in a CmdSet to produce the AREALIST shown in Figure 55 (page 148). Since the two variables are now in your dictionary you can eliminate the two DATASET commands from the CmdSet and use the remaining commands to produce the same AREALIST.

Editing and deleting new variables in your dictionary

You can edit the information about a dBASE variable that you added to your dictionary, by bringing up the dictionary window and highlighting the variable of interest. Click on Edit|Edit variable on the main menu bar and make the changes desired. Note that the short name is not available for change. Of course, editing the data, itself, must be done in your dBASE software.

A new variable can be deleted from the Dictionary by highlighting the variable in the dictionary and clicking on Edit|Delete on the Main Menu.
Saving DEFINEd variables in a winR+ database

winR+ provides the command keyword SAVE to record a DEFINEd variable permanently in the winR+ database. This not only saves the effort of copying commands for creating complex new variables from one CmdSet to others, but also allows the variable to be used afterwards with UNIVERSE which only accepts dictionary variables. Even more important, SAVEing some variables allows you to make multiple passes through the hierarchical database (see page 161). Finally, putting the variable into the winR+ database format allows for very rapid processing, of particular importance if the variable is frequently used and applies to, say, all persons in a country.

**winR+ file structure**

It is not necessary to understand the winR+ file structure to utilize all the program's features. However, since your SAVEd variables will appear as .bin files, you may wish to have an idea of why this is so. winR+ stores each variable of an entity in a separate file. For simplicity here, you may visualize the data for an entity, say, persons, as a matrix, with the rows as person records and the columns as the variables like age, sex, etc. The information on each person-row of the data matrix, might correspond to the values of the variables obtained from a questionnaire, with one questionnaire per person. In most statistical programs, each of these rows is a record within a single file of the data. In winR+, on the other hand, the data matrix can be seen in terms of its variable-columns, each of which is a vector of data. Seen this way, the first value in the sex column vector refers to the first person, the second value to the second person, etc. Key to winR+ is that each of the columns is stored as a separate file with the .bin extension—these are called transposed files. Using special files to keep track of the relationships between entities and the transposed files, this organization gives the system processing efficiency, since it allows winR+ to read only the variables specifically involved in a given process.

**Saving as a .bin variable**

After SAVEing a new or external variable as a .bin variable in your copy of the database, it is a variable like any other winR+ database variable and has the RANGE, TYPE, VARLABEL and VALUELABELS that you give the variable in the DEFINE; the latter two properties are
optional, but highly recommended. The new variable is associated with the winR+ database, but the new .bin can be stored in any directory that you indicate; it is recommended that you do not use the same directory as the original database.

To avoid "holes" in the data, the CmdSet creating the variable must be for the entire database, which must be done by using the SelSet ALL.

Compared with R+

The SAVE in winR+ is equivalent to GENERATE in R+. However, unlike GENERATE which requires the application of another module after having GENERAEd the variable, the Command keyword SAVE in winR+ takes care of the entire process with no intervention of the user beyond writing SAVE and the name of the variable.

Figure 58 illustrates how to SAVE the variable 90block.BlkNum. The variable created is the sequential number of the block within its district. The same algorithm can be used at any level, including for non-selectable entities. As will be seen (page 161), this seemingly useless variable is a very convenient way of identifying persons and other elements for use in later processes.

```
*01
RUNDEF Numblok
SELECTION all
*02----------------
DEFINE district.numblock
   AS COUNT 90block
   TYPE INTEGER
*03----------------
DEFINE 90block.BlkNum
   AS district.numblock
   TYPE INTEGER
   RANGE 1-1000
   VARLABEL "Seq BlkNum within distr"
   SAVE c:\winplus\workspace\BlkNum
*04----------------
TABLE numblok
   AS AREALIST
   OF 90block;
   90block.BlkNum
   OUTPUTFILE WORKSPACE numblok
```

Figure 58. Command Set to SAVE a variable that has been created to number the blocks within each district of the Selection Set.
Chapter 7. Additional Topics & Procedures — 155

Comments on the Command Set in Figure 58

Cmd01, the RUNDEF, has SELECTION ALL, since a SAVED variable must apply to the entire database. If you try to use a smaller SELECTION, you will get a compile error. UNIVERSE (and FOR in the DEFINE) can be used, since the resulting NotApplicable values are known and assigned to the records to which they apply.

Cmd02 counts the blocks within a district using COUNT. As each block is processed, COUNT adds 1 to district.numblock until all the blocks have been counted in a district, at which point it begins the count again with the blocks of the next district. The next command takes advantage of this counting action.

Cmd03 assigns the count to the block being processed. That is, as a given block is being processed, its sequential number is obtained by taking the count of the COUNT in the previous Cmd up to and including that block and assigning it to the block variable 90block.BlkNum, i.e., by DEFINEing it equal to district.numblock.

A RANGE and TYPE must be given. If it is TYPE REAL, the number of decimals must also be given, using OPTIONS DECIMAL d, where d is the number of digits to the right of the decimal. The VARLABEL must have no more than 40 characters, which is the maximum size of a label in the dictionary.

Finally, SAVED BlkNum gives the name BlkNum to the variable and its .bin file. To avoid confusion, it is recommended that the name be the same as used in the DEFINE, although it can be a different name. The SAVED variable name cannot be longer than 8 characters since it is the name of a file. For the same reason, no entity name is used; winR+ knows the entity level and includes it in the dictionary. If no path is given, the current work directory is used to store the .bin file of the variable.

Note that more than one variable can be SAVED in a CmdSet, using a DEFINE...SAVE command for each, and that external variables can also be SAVED to your winR+ database (if they apply to ALL the database).

Cmd04 is optional, but used here to show that the variable is correct. An AREALIST is created at the 90block entity level with the variable 90block.BlkNum. Viewing the AREALIST, it will be seen that the blocks are numbered sequentially and that the numbers start again when a new district starts.
When you SAVE a variable to the winR+ database as a .bin file, you are actually saving the variable to your copy of the database and its documentation to your copy of the dictionary in your Workspace. While you can give these to another computer, the master winR+ database for these data, of course, is not changed. Only the database administrator of the master database should make any changes to keep the master from being clogged with variables of little general interest and, more importantly, to ensure that only correct and well documented variables are added to the master database.

In addition, since the definitions of the new or external .bin variables are saved in your Workspace, if you create a New Workspace, these .bin definitions will not be included, unless you create the New Workspace from the previous one.

To delete a SAVEEd variable

Open the Dictionary and click on the SAVEEd variable to be eliminated. Then click on Edit|Delete in the main menu. Note that the CmdSet must be closed before beginning the delete.

Extending hierarchical processing in winR+

The first release of winR+ makes only one pass through the data during a given run; if there is sufficient demand for it, future versions may facilitate multiple passes through the data. However, in the first release there is a form of quasi pre-processing that can be used in special situations. Some examples of CmdSets that take advantage of this are given in the following section. Then in the succeeding section, suggestions are given for SAVEing variables for use in later processes to provide a type of virtual multiple-pass hierarchical processing.

Quasi pre-processing

From the previous chapter you are familiar with the use of COUNT in a DEFINE AS command to count the number of elements of a lower entity in the elements of a higher entity, e.g., the number of persons living on each city block in the SelSet of interest. The count can be conditioned by the use of FOR.
There is, in fact, another keyword, TOTCASES, that is also used with 
DEFINE AS and that also gives the number of lower entity elements in a 
higher entity, but that does not take into account any FOR conditions in 
the DEFINE command.

**Extreme caution must be exercised when using TOTCASES. While it 
makes possible the “pre-processing” required in the following 
examples, in other situations it can give “correct” results but that are 
normally not what you want!**

*This is because TOTCASES not only ignores any FOR condition, but 
also ignores the SelSet boundaries! That is, if you request the number 
of persons in the districts of the SelSet tutor2b (see page 42), you will 
get the number of persons in the entire San Antonio district, all of 
which is within the SelSet, but also all the persons in the entire Concepción 
district, even though only part of this district is within the SelSet!*

The advantages and disadvantages with TOTCASES are due to the fact 
that it is not obtained from a counting process. Rather its value for any 
given element of an entity comes from information — the position of the 
first and last record of each element — created when the winR+ database 
was generated and required for processing. Since TOTCASES is based 
on simple arithmetic, it is much faster than COUNT, but for the same 
reason can only give the *total* number of lower elements in a given higher 
entity element.

**Using TOTCASES for quasi pre-processing**

**Example of a FOR with 2 entity levels: Old persons living 
alone**

If you wish to know how many old persons by sex are living alone in an 
area of New Miranda, you need to know the number of persons in each 
house and whether the 1-person houses have a person that is, say, over 65 
years of age. You might think that one way of doing this is to process all 
the persons of the house with a COUNT to establish the size and get a 
frequency distribution by sex of those persons who are in 1-person houses 
and are over age 65. Unfortunately, this requires two passes, since as each 
person enters to be counted toward the total number of persons in the 
house, the person is also tabulated for the FREQUENCY. Hence, you 
cannot set a FOR condition involving both the age of the person being
processed and the result of a COUNT which requires that all the persons already have been processed in the house. One way around this quandary is to use the potentially dangerous TOTCASES as seen in Figure 59.

```
01
RUNDEF OIdAlone
SELECTION tutor2a
*02
DEFINE 90housin.90totpers
AS TOTCASES(90person)
TYPE INTEGER

*03
TABLE oldalone
AS FREQUENCY
OF 90person.90sex
FOR 90housin.90totpers=1 AND
90person.90age > 65
```

Figure 59. Command Set to determine the sex of persons 65+ years old and living alone. TOTCASES must be employed instead of COUNT to determine the number of persons in the house, since the former uses quasi “pre-processing”.

Comments on the Command Set in Figure 59

Cmd02 uses TOTCASES to determine the total number of persons in the house before beginning the one by one processing of each person in the house; the parenthesis must be used since TOTCASES is a function. Thus, the final value of 90housin.90totpers for the household being processed is known before the first person enters for tabulation in the TABLE AS FREQUENCY. As explained above, TOTCASES works very differently from COUNT, which if used here would count the persons in the house as they entered for processing, leading to erroneous results.

It is very important to note that the potential danger involved in using TOTCASES does not exist here because you are requesting a count of an entity 90housin whose elements are always included—or excluded—totally in the SelSet since 90housin is a non-selectable entity and we have no way of selecting only part of a house.

Cmd03 produces the desired FREQUENCY FOR the mixed entity condition in which 90housin.90totpers is, in effect, a variable that “pre-exists” before the FREQUENCY tabulation begins.
COUNT vs. TOTCASES

To emphasize that TOTCASES can only be used in special circumstances, such as in the above example, Figure 60 compares the total population of the two districts in the SelSets Tutor2a and Tutor2b. If you refer to pages 39 and 42, respectively, you will see that the former includes two complete districts, San Antonio and Concepción, while the latter has all of San Antonio, but only part of Concepción.

In each panel of the Figure, a FREQUENCY is shown for the total population at the district entity; the code is the value of the variable and there are two values since there are two districts (another form of presentation would be to show AREALISTs for each). When both districts are fully included in the SelSet, outputs A and B, the results are the same for TOTCASES and COUNT, namely, 1307 and 2086 persons in the two districts, respectively.

However, when the SelSet do not include complete districts, outputs C and D are different. The incomplete district has 983 persons in the SelSet as correctly reported by COUNT, while TOTCASES reports, again, the size of the total district, 1307 persons. Either might be what you really want, but you have to know what you want and remember the difference between the two keywords.

Finally, it should be noted that TOTCASES can only be used with dictionary variables and not with variables created during a run.

Remember also that TOTCASES cannot be used in a DEFINE that includes a FOR expression, since it will totally ignore the filter and give you erroneous results.
Figure 60. Population counts in 2 districts using TOTCASES and COUNT. The upper row, outputs A and B, are both for SelSet Tutor2a which has two complete districts, San Antonio and Concepción. The lower row, outputs C and D, are both for SelSet Tutor2b which has all the San Antonio district, but only a part of Concepción. As seen in C, TOTCASES gives the total population size (1307) of the district even though not all are within the SelSet, while COUNT gives the true population size (983) of the part of the district within the SelSet.

Thus, unless you are dealing with the special situations presented in the preceding two examples, or really want to obtain the TOTCASES for the elements of an entity even if not entirely within the SelSet—the information on larger area might be needed as the context in which the SelSet is inserted—it is strongly recommended that you do not use TOTCASES.

If you cannot resist the urge to work with TOTCASES, try to use it only with non-selectable entities (see page 23).
Virtual multiple-pass processing SAVE variable

As explained in the previous section, \textit{winR+} makes a single pass through the data and that, therefore, all its hierarchical processing is conditioned by this fact. In particular, it is impossible to use the results calculated at a higher entity in a hierarchical process to filter lower entity elements. Thus, a \texttt{COUNT} cannot be used to count the number of youths aged 12-19 on a block, and then this number used in the same CmdSet—that is, in the same run—to filter whether a given person lives on a block where there are few youths.

However, if the number of youths on each block is first \texttt{SAVE}d (see page 153), then it can be used in another CmdSet to filter persons according to whether they live on a block with few youths along with any personal characteristics that may be relevant to the analysis.

A triple multi-pass process

More elaborate multi-pass processes may be necessary. For instance, in the example of a CmdSet to determine the sex of the oldest-aged person in a household (see page 121), we were limited in the analysis that can be done, since we have located the oldest person, but the person may already have been processed by the time we know, for example, when the 3rd person is the oldest of the five in the house. In the CmdSet cited, a trick to get around this problem was given, but it is not convenient if we need many variables for an analysis and may breakdown if complex \texttt{FOR}, etc., are required.

One solution is to “identify” the person who is oldest, \texttt{SAVE} the identification, and then use it in CmdSets concerned with detailed analysis of the aged. A convenient identification is the sequential number of the person in the household. This can be obtained using the CmdSet shown in Figure 58 on page 154, but adapting it for persons within the household. After this variable, perhaps called \texttt{90person.NID}, is \texttt{SAVE}d to the database, a second CmdSet would determine the oldest person in the household, and \texttt{DEFINE} a variable such as \texttt{90housin.oldNID}, just as the sex of the oldest person was \texttt{DEFINE}d. This variable, \texttt{90housin.oldNID} would then be \texttt{SAVE}d to the database. Finally, a third CmdSet would create whatever analysis variables were necessary and produce the necessary output tabulations, always limiting the results to the person in the house who had their \texttt{90person.NID} equal to \texttt{90housin.oldNID}. 
Working in the AREALIST Browser

All generations of the Redatam software have been designed to facilitate the easy location of user-defined small areas from within a much larger dataset and the rapid processing of the data from these areas. Over the past few years, users have become concerned not only with the processing of a given area such as a municipality, but also with the processing and subsequent spatial analysis of sets of small areas, such as of each of the thousand or more blocks of a municipality. To meet this emerging need, winR+ has introduced the AREALIST as a specific type of output table along with a powerful browser to help you manipulate your lists of areas and their variables (fields) for such purposes as map display, GIS analysis, specification of target groups, etc. An illustrative AREALIST is shown in Figure 61.

This Section describes the use of various features of the AREALIST Browser in some depth; to bring all the information on the browser together, there is some repetition with that provided in the Tutorials and elsewhere.

The first release of the winR+ AREALIST Browser can only view AREALISTS stored in a winR+ Workspace. Consequently, AREALISTS that have been created, for example, as a .dbf file, must be imported into the Workspace to be manipulated by the Browser; import and export of AREALISTS is discussed starting on page 167.

To recapitulate, an AREALIST (see Figure 61) is a matrix of records and columns where:

- **Records or Rows refer to Areas.** The rows of the AREALIST are the individual records for each area. The code for each area, which are all elements of the same entity—in this case, of the 90Block entity—is given in the far left column.

- **Columns refer to Variables (Fields).** The columns of an AREALIST matrix each hold the values a given variable; sometimes these variables are called fields. A variable may be at the same entity level as the AREALIST record, such as the total number of persons, Totalpers, on each block in Figure 61. When the variable refers to a lower entity level, such as the sex of persons in an area, then the
variable in a given column is for a single category, such as sex1=male; 
the values of this variable give the number of males in each area.

The records of a given AREALIST are always the elements of a single 
entity level. For example, they can be blocks or districts, but not a 
mixture of blocks and districts. Furthermore, in the first release of 
winR+, the records are always elements of a selectable entity; thus, in 
New Miranda, the records never refer to persons or houses, since these 
are non-selectable.

It is important also to remember that an AREALIST created in winR+ 
always is for the territory defined by the specific Selection Set that was 
used in the Command File that created the AREALIST.

Overview of the AREALIST Browser

AREALIST name: The AREALIST name, 14b_al3a in the case of 
Figure 61, appears on the third line of the window within the scrolling 
business:  

Many of the following actions on Columns and Records use Record 
Columns on the Main Menu; these entries are available only when the 
AREALIST Browser is the active window.

Records

Changing row height: To change the height of all rows of an AREA-
LIST, place the cursor on the separation between any two rows of the 
left- most column of the Browser window; click and hold down the right 
mouse button. The cursor changes to a double-headed arrow and all rows 
expand or contract as you drag the cursor down or up, respectively.

Scrolling records: Use the Top and Bottom buttons, and 
respectively, to reach the beginning and end of the list rapidly.

Similar buttons with triangles alone alongside the Top and Bottom 
buzztons, can be clicked to move the line cursor Up or Down the records 
a row marker triangle in the left-most column moves up or down—. 
You can also use the window scroll bar(s). If they are not visible, the 
scroll bar(s) will appear after clicking in the body of the table; of course, 
if the list is short and fits within the window, the scroll bars will not be 
shown.
**Manual selection of a record (area):** Within the Browser, you can also move the row cursor to an area by clicking anywhere in its record. You can *Select* an area by clicking on the left-most column of the window; the record selected is highlighted by a change in its color. You can *Deselect* an area by clicking again in the left-most column.

The number of areas selected is indicated at the bottom of the AREA-LIST Browser window: \[ \framebox{4 selected from 118} \]. Clicking on Record|Select Record and Record|Unselect Record on the Main Menu are other ways of selecting and deselecting records.

This window also provides buttons, to move the line cursor *Up* and *Down* from selected area to selected area (ignoring the non-selected areas) and buttons, \[ \] and \[ \], to move to the *Top* and *Bottom*, respectively, of the set of selected areas.

For other methods of record selection, see the section starting on page 168.

**Selecting all records and Clearing all your selections:** If you want to Select all records of your AREALIST or get rid of all the selections that you may have made, you can click on Record|Select All or Record|Clear all Selections, respectively.

**Inverting your selections:** When you Invert a set of records, all the selected records are converted into unselected ones, and all the unselected into selected. Click on Record|Invert selection.

**Sorting the list of records.** You can sort the list of records by any of the column variables; select the variable of interest from the combo box at the top of the window: \[ \framebox{Sort by Natural order} \].

Natural order is according to the geographical hierarchy in the database, i.e., by the codes of the entity, in this case *90block*. The order can be made *Ascending or Descending* by using the first or second button with arrows at the right of the combo box, respectively.
Figure 61. AREALIST Browser with the AREALIST 4b_a13a with records of the entity 90blocks. Sorted in natural order, i.e., by block_4ba13a.wpg

Columns

Names of columns: Each column has a name which was assigned when the AREALIST was created. See page 104 for rules on the naming of AREALIST variables. When a column refers to a variable of a lower level entity than that of the AREALIST, such as the sex of persons on each block, as many columns appear as there are categories of the variable and each column has the name of the lower level variable with the code appended, e.g., sex1, where 1 = males and sex2, where 2 = females.

Changing column width: You can expand or contract a column width by placing the mouse cursor on the vertical line separating two columns; this must be done in the row of column names. The cursor turns into a double-headed arrow and the column can then be pulled narrower or wider. The change affects the column to the left of the cursor; the increase or decrease in column width is taken from the rightmost browser column. After adjusting all the columns from left to right, make the final adjustment in the right-most column by increasing or decreasing the right-side of the browser window.
Hide/Reveal columns

a) To hide a column: Click on a column title to select it (the title cell changes color). In the Main Menu, click on Columns|Hide Column and the column will disappear. Clicking on another column title and repeating the procedure will hide that as well. You also can use the Sort by combo box, to select each column to hide.

b) To show a hidden column: Use the Sort by combo box to select a hidden column. Then click on Columns|Show Column on the Main Menu.

Column statistics: Click on a column name, or select the column name in the Sort by combo box. Then click on Columns|Statistics on the Main Menu. A table appears with the column name (called here, the Field name) and a series of statistics on the column, including the a) Total number of records; b) Number of null records (those which are blank without any value; records with a 0 value are not null); c) Number of distinct values (only the first instance of a given value is counted, if more than one record has the same value); d) Minimum and maximum values which give you the range; e) the Average, the Sum of all the values, Standard deviation and the Median. These only apply to fields that are numeric, type Integer or Real.

Editing a column: Select a column by clicking on its name (or via the Sort by combo box). Then click on Columns|Edit; the column will change to a red color, which allows you to make changes in any record in the column. When you close the AREALIST Browser, your change(s) will be saved in the AREALIST.

Note that only columns of an “original” AREALIST can be edited. Columns that have been added to an AREALIST (see page 173) and then calculated cannot be edited without first saving the result and then opening the saved AREALIST.
The first release of winR+ does not have an UNDO function. Use the Editing facility with great care to make changes in an AREALIST, since you can distort your results if you make an error and you cannot remember the original value(s). If possible, print your AREALIST before beginning the changes or make a backup copy by Saving it under a different name.

Importing and Exporting AREALISTS

Import a .dbf into your Workspace as an AREALIST

Procedure

1. **Make the Workspace the active window**, Although not necessary, you may find it convenient to close all the AREALIST windows, if any are open. Of course, the records must correspond to areas of an entity in your winR+ database structure.

2. **Set the combo box of the Workspace to AREALISTS**.

3. **Click on File|Import|From xBase File** on the Main Menu.

4. **Select the .dbf file of interest** from the Choose xBase file... window. Click the name to select it and then click the OK button. The file will appear on your list of AREALISTS under the name of the file.

5. **Select the new AREALIST and Open it**, to see it in the Browser.

6. **To use a .dbf file as another AREALIST**, the key field must be defined. To do this, select the field and then the option Column|Set as Primary key on the Main Menu Bar.

Import AREALISTS from another Workspace: If you have another Workspace on your computer, you can import one or more AREALISTS from it into your current Workspace by clicking on File|Import|From Other Workspace and then selecting one or more AREALIST in the other Workspace that opens. As indicated in the instructions at the bottom of the Workspace, holding down the Control button on the keyboard allows selection of more than one AREALIST.
Export an AREALIST from your Workspace

You may wish to export an AREALIST to a .dbf file to use in software that reads .dbf files, such as an spreadsheet package as Excel or Quattro Pro.

Procedure

1. Open the AREALIST to be exported. If there are already AREALISTs open, make the one to be exported the active window.

2. Click File|Export to DBF on the Main Menu.

3. Select a directory and give the .dbf file a name in the Export as ... window that opens (it is recommended that you use the same name as the AREALIST, which is the default provided by winR+). If you do not want the dBASE III file format, you can select another variation of .dbf in the combo box at the bottom left of the window. When you are ready, click the OK button.

Selecting and saving records of an AREALIST

You can select individual records manually from an AREALIST, taking advantage if you want, of the sort facilities to assist the selection — e.g., selecting the top or bottom 5 areas on a given variable (see page 164 for an outline of manual selection). winR+ also provides a powerful Query (SQL) facility, which allows you to build an expression to logically select sets of records or areas that meet a criteria. If you want, you can then combine sets obtained from different criteria.

The Query Expression Builder for record selection

The Expression Builder appears in various guises in winR+. In the Easy-Tabs Simple Count, it can be used to create an expression to condition items that are counted. When creating Command Sets, the UNIVERSE expression of the RUNDEF command can be created with an Expression Builder, as can the FOR expressions in the DEFINE Assist and the TABLE Assist, respectively.

General use of the Query Expression Builder to select AREALIST records is outlined in the following procedure that uses Figure 62 on page 171 as an example. The AREALIST for the 43 districts of New Miranda is sorted by the variable AvperHou which is the average number of persons per house in each district. The objective of the example is to
select all districts which meet both of two conditions: a) the average number of persons per house is less than 5; and b) there are at least 200 houses in the district (variable TotHous). Thus, we want to select districts evaluating True for the expression: AvperHous < 5 AND TotHous > 200.

Procedure

1. **Make the AREALIST the active window.** This must be an AREALIST from your current Workspace; if it is an external file, it must be imported into your Workspace (see page 167).

2. **Access the Query Expression Builder** by clicking on the Main Menu Record|Select by Query. The second line of the Window shows the path and name of the AREALIST for which the Query will be created. Note that the AREALIST AL_Dist1 is a table of the Workspace (WMIRI.mdb) and therefore is written WMIRI.mdb|AL_Dist1.

3. **Create the expression** in the large box above the buttons of Operators. You can write and erase in the expression box from the keyboard and use the cursor as a place marker. There are facilities to help to do the writing for you:

   - **Calling up Fields:** In the first of the three bottom list windows, click on Fields, as the AREALIST variables are called here, to show the Fields in the second list.

   - **Add a field to the expression.** Double click any field to transfer it to the expression at the point where you left the cursor.

   - **Add an operator or parentheses to the expression.** Single click on the operator button to place the operator where you left the cursor. You can call up more operators by clicking on Operators in the first list and then clicking on All in the second list and double click the operator of interest in the third list.

   - **Values of a field.** Clicking on a field in the second list, shows its sorted values in the third list. The small combo box above the third list allows you to see All values or only those that are Distinct, i.e., if there are any sets of two or more records with the same value, only the unique values are shown.

   - **Add a value to the expression.** Double click any value to transfer it to the expression—or write it from the keyboard—.
4. **Check the expression.** When you have the expression ready, verify that it does not have syntax errors by clicking on the **Check** button. If there is an error, it will be reported in the bottom line above the **Check** button. If there are no errors, the number of records selected will be indicated (but they will not yet be highlighted on the **AREALIST**). In this way you can check your expression as you finish each logically completed part and you can change the expression if it is giving you too many or too few selections.

5. **Highlight the selected records on the AREALIST,** by clicking on the **Add to selection** button. This applies the expression to the **AREALIST.** If there was a previous selection (manual or by Query) this adds to the records selected. Previously selected records remain selected.

6. **To replace selected records with another Query:** After finishing the new expression and checking it, click the **Replace selection** button; this erases all previous selections and replaces them by the new selections.

7. **To start all over:** Click on **Record|Clear all Selections** on the Main Menu with the **AREALIST** the active window.

8. **To temporarily store a selected record set, while looking at another set:** To avoid losing a set of selected records highlighted on your **AREALIST,** when you want to try another Query, you can store the former in a **temporary set** by clicking on **Record|Store in Temporary Set|Add to Temporary Set.** If the temporary set already has selections, you have the option of **Adding to or Replacing them.**

The number of selected records in the Temporary set appears in a box at the right bottom of the **AREALIST** window.

Since you cannot save the Temporary Set in the first release of **winR+,** if you want to use it, you must click on **File|Restore from Temporary Set** on the Main Menu and decide whether to Add the Temporary Set to current selections or Replace them.

9. **Show only the selected set of records in the AREALIST:** To help examine its content or to save only the selected set, click on **Record|Show Selected Only** on the Main Menu with the **AREALIST** as the active window.

If you have only the selected records visible, to continue adding more records, you must first click on **Record|Show All.**
Saving the selected records in an AREALIST

When you have completed your selection of records based upon the data processed for the territory that you initially defined in your Selection Set, you can use the selected records, for instance, to create a new Selection Set (see page 177) that defines your target population more adequately, and then use \texttt{winR+} to re-process data for that sub-territory alone. However, for other purposes—such as viewing the selected areas on a map (see page 192)—you may wish to save the selected records as a new AREALIST using the following procedure.

Procedure

1. **Eliminate the non-selected records.** Click on Record|Show Selected Only on the Main Menu with the AREALIST in the active window.

2. **Call up the Save Table As window** (see Figure 63) by clicking on File|Save As on the Main Menu. Fill in the required information:

   - **Select the AREALIST type** in the top combo box. In this release of \texttt{winR+}, it must be a \textit{Workspace Table}.

   - **Define a name and label.** The two should be separated by hyphen as shown in Figure 39. The name should have no more than 8
characters. If it is to replace a previous table, i.e., AREALIST, in the Workspace, you can choose the name in the combo box. This gives you a way of changing the label of an existing table.

- **Select the fields to be included.** Select/deselect as usual by clicking in the left-most column of a row and use the buttons at the bottom to assist. The key field with the area code, here **district**, must be included. In the example, @distr, is excluded.

- **Change the names of any of the fields.** This is done in the **New Name** column of the field selection box.

  Field name changing is optional and allows you to make improvements in the names. However, it must be done in some cases when making an AREALIST Save As since a field name must start with a letter and must have 8 characters or less.

3. **Make the Save.** The new AREALIST will appear in your Workspace.

As this procedure can be used to Save As any AREALIST in your Workspace it can be used to change a name or label, deleting the old version.
Figure 63. Window for making a Save As of an entire AREALIST or the selected records of an AREALIST. The fields to be saved can also be selected, but must include the primary key, here \textit{district}. 

\section*{Adding new columns (variables) to an AREALIST}

To facilitate the selection of records or for later display on a map, etc., you may wish to create one or more new columns so that you can define a variable (field) for each.

\textbf{Procedure}

1. \textbf{Open an AREALIST} from your Workspace.

2. \textbf{Create a new column in your AREALIST} by clicking on \textit{Columns|Add Column} and providing a column name. The new column will be added at the right-most side of your AREALIST. The original columns will have the \textit{AREALIST} name added as a prefix, and the new column will have this and the word \textit{Extension} as a prefix to the column name that you assigned.

3. \textbf{Calculate the values for each record of the new column.}
   Click on \textit{Columns|Calculate Column} on the Main Menu. You will
then be given another sub-menu with a choice of two methods of calculation: **Expression** or **Classification**.

3a **Expression:** An Expression Builder opens to allow you to calculate expression such as adding two existing fields, etc. (see page 168 for general information on the Expression Builder).

- **Erase any information** that is initially written in the expression box.

- **Write the expression and enter it:**

  **An expression for all the records:** If the formula applies to all the records (such as adding together values of other existing variables for each record), write the expression and then press the buttons in sequence: **Select records**, **Compute expression**, and, if you are satisfied, **Done**. It is recommended that you check a few values, since even a correctly written expression may not be substantively what you want. If there is an error, you can start again.

  **Expressions for a sub-selections of records:** If you use a **True/False** expression first to select some of the records —e.g., **Sex1 >300**, where **Sex1** is an existing variable— and press the **Select records** button, the bottom line will tell you how many records meet the condition. Then erase the **True/False** expression and replace it with one that assigns a values to the selected records; this can be a combination of values of existing fields (e.g., **Sex1 + Sex2**), a constant, etc. Repeat this procedure as necessary for the remaining records.

3b **Classification:** The **Classify** window opens (see Figure 64). **Classification** in this situation involves dividing the values of a variable into groups and then assigning a number to each group; this is similar to classification for assigning colors to a variable prior to mapping its distribution (see page 187).
Figure 64. The window to calculate the values for a new column of an AREALIST by classifying an existing field. In this case, the existing field `totmales` has been classified into 4 classes such that the number of records in each is approximately the same (Quantiles). The minimum and maximum is given for each of the classes. (si:ALClas.wpg)

- **Select an AREALIST field** from the list provided in the upper right combo box. Note that the fields have the name of the AREALIST as a prefix.

- **Decide on the classification type** that you wish to use from the second combo box on the right.

  **Quantiles** (shown in Figure 69): winR+ divides the range of the field into the number of classes chosen such that there are approximately the same number of records in each class; the classes are assigned integer values. This is frequently used since it ensures a reasonable number of records in each class.

  **Equal intervals**: winR+ divides the overall range of the field into the number of classes chosen such that each class has approximately the same interval or range. The number of records in each depends on their distribution and there will be situations with highly skewed numbers of records in each class, i.e., some classes will have many cases and others very few.

  **Manual**: The user divides the range into “arbitrary” intervals based on theory or convenience; the intervals, in turn, determine the numbers of records in each.
• Decide on the **number of classes of the field** using the box provided. Too few will throw away information, but too many will result in few records in many classes and give uncertain results.

• **Check the result** by clicking on the **Classify** button; this will show you the integer value assigned to each **class**, the **count** = number of records in each class, and the **minimum** and **maximum** value for each class. This button does not actually assign values to the column, so you can easily make changes.

• **Use the classification scheme to assign the values** to the newly created **AREALIST** column by clicking on the **Done** button.

If you do not like the result, you can repeat the procedure making changes. Note also that you can add various columns to an **AREALIST**, following the above procedure for each.

**Linking two AREALISTS**

The final operation on **AREALISTS** that will be outlined involves linking two tables, i.e., **AREALISTS**, from the same Workspace. Of course, for such a link, each table must refer to the same geographical entity—such as **districts** or **90blocks**, etc., in New Miranda—. The codes of the geographical entity are called the **key field** or sometimes the **primary key**. So that there are records (areas) to link, the two **AREALISTS** must have some of their areas, and hence, codes, in common for each of their key fields.

**Procedure**

1. **Open an AREALIST from your Workspace.** This will be called the **primary table** or **AREALIST**.

2. **Open the Link Table window** (Figure not shown) by clicking on File|Link to... on the Main Menu.

• **Select the AREALIST to be combined** from the combo box, which lists all the **AREALISTS** in your Workspace. This is the **secondary table**. If you want to use an external table, you must import it into the Workspace before beginning this procedure.
Chapter 7. Additional Topics & Procedures — 177

- **Select the fields to be attached** from the secondary to the primary table by double clicking on each field in the Field list. When you do this, make sure to include the key field from the secondary table. If you make a mistake, double click the selected field to deselect it.

- **Indicate the keyfield in the secondary table** by highlighting it in the Selected Fields and then clicking on the triple arrow button; it will be taken from the list to the box. If you choose the wrong field as the keyfield, it can be erased by highlighting it and erasing it with the backspace key.

- **Make the transfer** by clicking on the OK button.

3. **View the new AREALIST.** The new combined AREALIST will appear with the name of the primary table. Each column field will have the name of the field with a prefix of the name of the originating table.

   Note that the new AREALIST has only those records that match the keyfield of each of the input AREALISTS. Records in the primary or secondary AREALIST that do not match on the keyfield are eliminated.

4. **Save the new AREALIST.** Follow the Save As procedure on page 171. Be careful to observe the rules on field names, making any necessary changes in the new name. If there are duplicate names, change at least one of them.

Deriving a Selection Set from an AREALIST

**Creating Selection Sets using computed criteria**

Variables can be included in an AREALIST to use in making a sub-selection of the areas of a SelSet. For example, one might want to identify the blocks of city that have 50% or more women household heads. Sorting an AREALIST with this variable, would allow the researcher to identify the blocks meeting the 50% or more criteria. The codes of the blocks could then be used to enter a new SelSet. In fact, winR+ automates
this process so that once the individual sub-areas are selected (manually or by a Query), you can request that a new SelSet be generated.

In the following sections, three ways are given to generate a SelSet:

- Using semi-manual selection of areas in an AREALIST.
- Using the Query Expression Builder in an AREALIST.
- Using TABLES...AS AREALIST, with OUTPUTFILE SELECTION in a CmdSet.

SelSet from semi-manual identification of specific areas

As an example of how to create a new SelSet from an AREALIST, we shall use as an example the AREALIST at the 90block entity level for the SelSet Tutor2a created with CmdSet T4b_ALA in Tutorial 4b (see page 100). We then require a new SelSet that includes only the blocks with a large average number of persons per house, 90block.AvPerpHou, e.g., with the number of persons per house greater than 6.

Procedure

Refer to Figure 65 which shows the various windows involved in the following procedure.

1. **Open the AREALIST in your Workspace.** Note that if you were to start with a .dhp file, you must import it into your Workspace and then open it — of course, it must have the appropriate key field, here blocks of the 90block entity.

2. **Sort on the variable of interest,** here, 90block.AvPerpHou, using the combo box at the top of the AREALIST window. Since we want to see the blocks with a high average house size, you can use a Descending Sort (second of the buttons to the right of the combo box); the blocks with the largest average will be at the top.

3. **Manually select the blocks (records) of interest,** i.e., on each of those with 90block.AvPerpHou > 6, by clicking on the left-most edge of each. The seven records selected will each become highlighted. If you make a mistake, you can click again to remove the selection. All selections may be cleared by clicking on Record|Clear all Selections on the Main Menu.

4. **Create the SelSet** by clicking on File|Create Selection File on the Main Menu. You will be asked for a name and label for the SelSet;
any name up to 8 characters can be used, but T4b_ALA is might be preferred to facilitate easy location of its origin—it will not be confused with the CmdSet, since each is a document in a different list of the Workspace—the SelSet will be created in your Workspace.

6. View the new SelSet, by opening the SelSet T4b_ALA in your Workspace. If you have carried out this example, you will see, as shown in Figure 65, the usual tree structure with the selected blocks with filled black circles. Of course, if you want, you can edit the new SelSet in this window.

Figure 65. AREALIST resulting from the Tutorial 4b Command Set (see page 100). The screen shows the AREALIST for the SelSet Tutor2b by blocks in descending order by average number of persons per house. The 7 blocks with over 6.0 persons per house have been selected manually (or by Query). Part of the resulting new SelSet that has been created from this selection of the 7 blocks is shown in the Selection Editor.

With the winR+ map module, you can create a SelSet directly from a map or from a map linked to an AREALIST; see page 196 for a figure illustrating this and associated explanatory text.
Specific areas using the Query Expression Builder

When the identification of specific areas of an AREALIST involves quantitative criteria, as in the previous example, and particularly when the criteria are complex and the number of areas large, you can use the Query expression builder. As this is discussed in detail on page 168, the following procedure briefly outlines how to accomplish the same result by Query as was done manually in the previous section:

Procedure

1. Click on Record|Select by Query, with the AREALIST as the active window. The window shown in Figure 66 has an Expression Builder (SQL) that allows you to define an expression in the large window that is then used to select the areas.

2. Create the expression: Select Fields in the combo box on the left below the operator buttons, and then from the list of fields in the AREALIST, select avperphou and click on it twice to send it to the expression window. Click in the expression box for > and type in a 6.

3. Click on the Check button, to see how many areas have been selected. In this case, the number should be 7, as in the semi-manual example.

4. Click on the Add to selection button to make the actual selection of the areas in the AREALIST.

5. Close the Query window with the Done button.

6. Create the SelSet, by clicking on File|Create Selection Set and proceed as in the semi-manual case.
Generating a SelSet directly in a CmdSet

If you want to create a new SelSet based on fairly simple quantitative criteria and do not need to view the areas selected via the manual or Query procedures before making the SelSet, you can generate the SelSet directly in a CmdSet.

Repeating the same example again, in CmdSet *T4b_ALA* (see page 100), in the TABLE AREALIST command, rewrite it so that the command appears as:

```
*07-----------------------------
TABLE t4a_2
AS AREALIST
OF 90block,
  90block.totpers,
  90block.totfem,
  90block.totmales,
  90block.totHous,
  90block.AvPerpHou 2.2
OUTPUTFILE SELECTION T4b_ALAs
FOR 90block.AvPerpHou > 6
OPTIONS
  TITLE "Selection created by CmdSet to incl blocks with AvPerpHou > 6"
```
The lines that have been changed, are shown with a larger font. To avoid overwriting previous work, note that the SelSet is named \texttt{T4b\_ALAs}. In addition, the entire CmdSet has been given the same name, \texttt{T4b\_ALAs}.

As with an AREALIST, after running the CmdSet and clicking on the AREALIST hypertext in the Report, you can view the Selection Set with \texttt{View\textbar{}Table Presentation} on the Main Menu. This takes you directly to the Selection Editor. In this example, the result should be the same as obtained via the manual and Query procedures.

You can also create a Selection Set directly from a map with the mapping module of \texttt{winR+}; see page 194.

\section*{Using an external Selection File}

A CmdSet can use an external DOS Selection File (extension \texttt{.sel}) which meets your needs — the file might be a Selection from \texttt{Redatam-Plus} or it could be a Selection File previously exported from a \texttt{winR+} Workspace.—

If you want to use the external Selection file \texttt{plan.sel}, the \texttt{RUNDEF} command will have the clause

\begin{verbatim}
SELECTION SELFILE c:\proj\territ\plan.sel
\end{verbatim}

which must include the full path to the Selection File.

Alternatively, you could go to the Workspace window and set the Document Type window equal to \texttt{SelSet} and then click on the Main Menu File\textbar{}Import Document\textbar{}From DOS File to bring the Selection File into your Workspace. Then you can use the SelSet as usual in your CmdSet.
Using AREALISTs with the winR+ map display

winR+ is available with a map display module which permits 1) digital maps to be brought into the winR+ Workspace; 2) the classification of a AREALIST variables that assigns a different color to each category or sub-range of values; 3) the display of the classified variable on the map. In the first release of the system, digital maps can only be brought into winR+ in the .bna format of Atlas GIS or the MapInfo export format .mif/.mid; digital maps of some other GIS can be exported to these formats.

Adding a map definition to your Workspace

To illustrate how to work with the map display features of winR+, we shall outline the procedure for adding the map definition of a district level map of New Miranda to your Workspace. Atlas GIS format files describing the district polygons along with the codes that connect the polygons to the database districts are in file wmir.bna in the directory where you installed it such as c:\winrplus\wmirmap or c:\winrplus\work, or wherever you placed it when you did the winR+ installation.

Procedure

1. Set the document type to Map Definitions in your Workspace combo box.

2. Open the Map Definition Editor window by clicking on the New button of the Workspace window or File|New on the Main Menu. The editor is shown in Figure 67.

2a Select the GENERAL Tab.

- Define a short map name. The map name can have up to a maximum of a 8 characters.

- Indicate the entity used to define the map, in this case district. This is the key that will link to an AREALIST key.

- Select the Geographical Type from the combo box. For this example, select the Atlas GIS format .bna.
- Indicate the **Data Source of the map**, which is the .bna file that can be found with the help of the **Locate** button.

- **Set the Coordinate Conversion** to SW. If the map that results is inverted, you can try other settings; the Western Hemisphere south of the Equator is SW.

Do not click the **Done** button until you have completed the second Tab

2b **Select the second Tab.** Sets defaults that can be changed when making a specific map.

- **Set the Map font.** You can set this for the screen only or for printing and the screen. In the first release of winR+, the fonts cannot be changed here.

- **Set the Default color set.** Double clicking on a color box, brings up the **Select a color ramp** window. Double click on a color to select it on the ramp (see page 189 for more information on color ramps.

  The “Default” color is the color of the base map. Note that the “selection” color is used for highlighting on the map and should be the same color used for the AREALIST record selection.

4. **Complete the map definition** by clicking the Done button.

5. **Display the map** by clicking on this Map Definition in your Workspace and then on the Map Display icon . If the map is not positioned as you wish in its window, click and hold down the **right** mouse button on the map and drag the map to where you want within its window.

The map that appears (see Figure 67) has only the district borders since you have not yet connected it to an AREALIST to display the values of a variable. If you need to return to the Map Definition Editor at any time, select a map definition document in your Workspace and click the **Open** button.
Linking an AREALIST to a map definition

You can link any AREALIST and Map Definition at the same entity level and within the same Workspace — later versions of winR+ may also allow the linking maps with AREALISTS which are external dBASE or Microsoft Access files —. Since simply passing information from an AREALIST to a map can be done more easily — see the next section on classification —, the link procedure is done mainly to check consistency between a map and an AREALIST.

To illustrate the Procedure, an AREALIST for all New Miranda will be linked with a map for the entire country, both at the district entity level, in particular, map definition MapDist1 and AREALIST AL_Dist1 will be linked. This AREALIST, which is for all districts of New Miranda, can be obtained easily by making some simple changes in the CmdSet used in Tutorial 4b (see page 100): change the SelSet to ALL and entity 90block to district for all the variables; then SAVE AS AL_Dist1 and run.

Procedure

1. Open the Map Definition and AREALIST to be linked, by highlighting the Map Definition of interest in the Workspace, here MapDist1, and clicking on the Map Display icon. Then open...
an AREALIST from the Workspace, here *AL_Dist1* — as usual, this is done by clicking on the **Open** button after highlighting the AREALIST of interest —.

2. **Open the Link Forms** by first making the Workspace window active and then clicking on **Database** \(\text{[Link Forms...]}\) on the Main Menu. The **Link...** Window and the other relevant windows are seen in Figure 68.

- **Indicate the AREALIST to be linked:** In the upper combo box *current active*, select the open AREALIST. If you have used the same directories as here, the complete path to the AREALIST is: 
  \(c:\\text{winrplus}\\text{work}\\text{wmiri.mdb}AL\_DIST1\text{[Table]}\). This shows the AREALIST as a table in the Workspace *wmiri.mdb* — the Workspace is a Microsoft Access *mdb* database —, to which the name of the AREALIST is appended as a “Table”.

- **Indicate the Map to be linked:** Select the map from the *To* combo box, here *mapdis1 [MAP]*.

- **Establish the link**, by clicking the **Establish a double link** button and then the **Done** button. The link is double, because clicking on an area on the map highlights the corresponding row of the AREALIST and vice versa (see Figure 70 discussed below).
Figure 68. The top-left window defines the link between the AREALIST and the Map, shown in the bottom-left and upper-right windows, respectively. Part of the Workspace is also shown.

The AREALIST to Map Definition link cannot be saved in the first release of winR+. Fortunately, as you will see in the next section, it usually is not necessary to use the link.

Classifying a variable and displaying it on the map

If you want to examine the spatial distribution of the values in any AREALIST, you must begin by transferring the values to a map with the same areas as the AREALIST. This can be done directly without going through the link procedure of the previous section. However, in all cases, you must convert the values of the variable into classes — colors in a procedure called classification.

In effect, classification involves dividing the set of values of a variable — whether real or integer — into a set of categories or groups according to some principle, such as equal number of cases in each, or equal intervals between the minimum and maximum value of each class, etc. Once the classes have been defined, colors can be assigned to each group. Normally, colors seem to have a visual scale from light to dark, so that lighter can be taken to mean less of the variable and darker more.
Classification of a variable

The following procedure assumes that you have the AREALIST. It is preferable that you start the classification procedure with only the AREALIST of interest open; and map of interest on the screen and have already created the link. When you have completed the classification procedure and open the map, the link will be created automatically.

Procedure

1. Make the AREALIST the active window.
   
   - Select the column with the variable of interest by clicking on the column; alternatively, you can select the variable in the Sort by combo box.

3. Open the Classification and Mapping window by clicking on Columns|Link to Map on the Main Menu (see Figure 69). The AREALIST name will be shown along with the entity level, here district. Click on each of the 4 Tabs in order, and complete the information before using the Save and Exit buttons.

3a GENERAL Tab (Figure not shown) click it if it is not the active Tab.
   
   - Select the variable (field) to be classified from the combo box on the upper right (see Figure 69), here AvPerPHou. The grid on the left shows the individual values of the variable for each of the areas (districts, in the present example) and the box on the bottom right gives you statistics for the variable such as the average, median, minimum and maximum values of the range, etc.

3b CLASSIFY Tab.
   
   - Indicate the classification type to be used. Each class is assigned an integer and a color for the map.

   There are various ways of dividing the data for a variable into the number of classes that you request; winR+ provides the following types to choose from:

   Single value. Each value is a class of a different color.
Quantiles. The classes are automatically defined such that each has approximately the same number of cases. This is the type shown in Figure 69.

Equal intervals. The range of the variable is divided into equal intervals, which in turn determines the number of areas in each class; if the distribution on the variable is very skewed, there may be many areas in one class and few in the remaining classes, which is inconvenient for most analysis.

Manual. You set the minimum and maximum for each class after defining the number of classes that you want.

None. This gives single values, but assigns a sequential number to each class, rather than using the values of the variable.

When there are minimum and maximum values in each class, note that the highest value in a class is up to and including the maximum value (e.g., 3.95 in the first class of this example) and the lowest value of the next class is greater than the minimum value (again, here 3.95); thus the apparent overlap between two classes is avoided.

- **Indicate the number of classes** (categories) into which you want the values of the variable classified. While the number of classes will depend on your substantive concerns, normally, it should be small; 3 is used in this example. Note that the number is ignored if your classification type is Single Values or None.

- **Carry out the classification** by clicking on the Classify button.

- **Select the Start and End colors.** Double click on the Start color to bring up the Select a Color Ramp window (see Figure 69). Choose the color ramp you want in the combo box and then double click a color. Do the same for the End color.

  You can eliminate this step if you are willing to accept the default start and end colors that you defined when creating the map definition (see page 184).
It is convenient to use the alternate color ramps for the actual classes to be mapped, since they each have tones of a single color which gives the effect of a visual scale from low to high. Normally assign the lightest color (left end of a ramp) to the first class and the darkest color (right end of a ramp) to the highest class.

- **Fill in the label** of the map legend by using one or more of the **Class**, **Ranges**, or **Counts** buttons. The **Clear** button erases the label information if you want to change it. You can also write in any texts that you want.

Do not use the **Apply** button until you have completed the rest of the Tabs.

3c. **MAP Tab** (Figure not shown).

- **Indicate the map**, using the combo box and then use the **Display** button to bring up the map, automatically creating the necessary link.

- **Write in or change the map text**: **Title**, **Subtitle** and **Comments** that you want to appear on the map display. These each appear on the map if the corresponding box is checked by clicking; another click unchecks a box. It is recommended that this information be shown, particularly if the map is printed.

3d. **LEGEND Tab** (Figure not shown)

- **Indicate what you want on the legend** and whether you want it displayed or not via the check box.

- **Select a colors requested**. The Default color is color of the entire map before displaying a variable or highlighting. Note that these color were chosen during the Map Definition; you can change them for this map, if you wish. As on the **Classify Tab**, double clicking the color box brings up the Color Ramp.

3e. **FONTS Tab** (Figure not shown).

- **Set the Map Font**, by clicking the button and making your choice. This is the font that will be used for the Title. All the other fonts in the relative font size box are in n tenths of the
Title font. Hence, if you write in 6 for the legend font, it will be 6/10 as high as the Title font.

4. **Apply, Save and Exit.** If you go back to a Tab to change any information, you can use these buttons on the last Tab where you are. The resulting map is shown in Figure 69.

![Figure 69](image.png)

Figure 69. Classification of the variable AvPerPHou from the AREALIST (upper left). The Classification and Mapping window (lower left) shows the variable divided into 3 Quantiles; the cursor is on the End color which opens the Select a color ramp (long bottom window). The resulting filled-in map is in the right bottom window.

7. **Printing the map.** As with any other winR+ output, you can print the map. Click on **File|Print** on the main menu. You can print in color if you have a color printer and set it for color. If not, you
should try to use colors that differentiate well in tones of grey and black and white.

As noted above, a variable from an AREALIST for a SelSet that has, for example, some complete and some partial districts, can also be displayed on a district level map. The value calculated for the partial district will be displayed, as a color, for the entire district. Hence, care must be exercised when interpreting the resulting map.

Displaying selected records from an AREALIST on a map

You may wish to view the spatial distribution of the areas (records) that you select on an AREALIST. If you want to work on a blank, uncolored map, bring up the Classification and Mapping window by selecting any numerical column on the AREALIST and clicking on Column|Link to Map on the Main Menu Bar; then go directly to the Map Tab to select and display the map. Any records selected on the AREALIST will be highlighted on the map and vice versa, and similarly for deselections of records or areas.

If you want the see the values on the map, you can follow the full classification and map display procedure of the previous section. Note that as each selection is made, the color of the area will change to the highlight color; deselecting an area restores its class color.

If you did not have the map available when you selected the areas on the AREALIST, bringing up the map to show existing selections does not work, but there is a simple trick. Show only the selected records and do a Save As (see page 171) to create a new AREALIST. Then follow the procedure to classify and display a variable on a map (described in the previous section), using a map for the original Selection Set or larger. Select any variable and classify it into, say, two categories by quantiles. When you define the Start and End color, use any color, but the same color, for each and apply the “classification” to the map. All the selected areas will appear “highlighted” on the map; areas without records in the AREALIST will remain uncolored. A lighter color will leave the boundaries of adjacent areas visible.
Editing the map objects and layout

A map display consists of the actual map, itself, a legend, title, sub-title, etc., each of which is treated as a separate object by winR+. You can change text or eliminate the title, legend, etc., of a map linked to an AREALIST by selecting any column in the AREALIST and then using the Classification and Mapping window (click on Column|Link to Map on the Main Menu).

In addition, you can change the positions of the objects in the map window (and on the printout) by positioning the cursor over the object of interest and clicking and holding down the right mouse button to drag the object to wherever you wish (but always within the map window); when the right mouse button is held down, the cursor changes to a four-way arrow indicating that the object can be moved in any direction.

Selecting and Querying areas on the AREALIST and map

Select Mode

After an AREALIST and a Map Definition are linked, whether or not a variable is classified, you can make the map window active and click on View|Select Mode (or View|Query Mode; see below) on the Main Menu; this permits you to click an area on the map causing it to be highlighted (normally a bright yellow) both on the map as well as on the AREALIST and vice versa. This is shown for districts in Figure 70. Clicking again on a district in either the AREALIST or Map, eliminates the highlight. If a variable is displayed on the map, the highlight eliminates the color for an area; clicking again restores it.
Figure 70. After linking both the AREALIST and Map shown, clicking in the left edge of a row (district) in the AREALIST highlights (selects) the row and the corresponding district in the Map window and vice versa. Clicking on a highlighted area in either window, removes the specific highlight in both windows.

Query "Point and Shoot" Mode

With the Map window active, if you click on View|Query Mode on the Main Menu, the cursor will change to an arrow and question mark as you move it over the areas on the map; double clicking to “point and shoot” an area shows information about the area on the status bar at the bottom of the screen.

Creating a Selection Set by clicking areas on a map

As seen from the above sections, you can identify areas on a map by various means and after identifying them, say, by classifying a variable, you can select them by highlighting. In addition, you can build expressions to identify areas by making the AREALIST the active window and then clicking on Table|Query in the Main Menu to bring up the Expression Builder; this Figure not shown here since it is the same as described on page 180. The result of a process of identifying areas of interest is shown in Figure 71, where the grey areas have been selected on the map with the corresponding district rows on the AREALIST highlighted.
Whatever the means used to select areas on the map, including simply clicking on areas of interest on the map, you now can create a SelSet for later use in a winR+ Command Set.

Procedure

1. Make a Map window active. If you have an open map with a variable displayed, close it, select it again in the Workspace so that the Map open icon comes up, and then click the map icon to bring up an uncolored map. Alternatively, you could link your map to a suitable AREALIST so that you can make selections either on the map or in the AREALIST with a reflection of each selection on both.

2. Create a Selection File by clicking on File|Create Selection File in the main menu. A Save Document As... window will open so that you can give the new Selection Set a name and label. The Selection Set created in this way—using a linked AREALIST (Figure not shown) to help make the selection on the map—is shown in the left window of Figure 72.

The SelSet, which will appear on your list in your Workspace, can now be used as any other, such as in a Command Set or editing in the Selection Editor.
Moving beyond winR+ via the winR+ Data Link

For many users, winR+ will often be the means to obtain complex geo-referenced measures that they require for analysis or modeling in a specialized package of their choice. The other package might be a Geographical Information System (GIS) for producing a more elegant and detailed spatial display than available in the map module of winR+, or for making sophisticated analyses and running models to aid spatial decision making in a particular field (e.g., see the R+GIS tools, page 9). The other package could also be a database system for elaborate statistical analyses.

In all these situations, the aim is to move data from an AREALIST source produced in winR+ to a file used by the target system. To this end, winR+ provides a number of procedures including a simple export and the winR+ Data Link.

For purposes here, we shall assume that the target system uses a .dbf file; future versions of winR+ will support a number of other formats as well, such as the .mdb of Microsoft Access.
Chapter 7. Additional Topics & Procedures — 197

Exporting an AREALIST in .dbf format to an external system

If you wish to relate the AREALIST to a .dbf in the external system and are not concerned with a permanent append or ensuring consistency (see below) between the winR+ output and the external .dbf, you can simply export the AREALIST from your Workspace as a .dbf to be used by the target system. In this case, the R+ Data Link need not be used.

As outlined on page 107, open the source AREALIST in your Workspace, make sure that it is the active window, and then click on File|Export on the Main Menu. The Export as... window will open to allow you to select the output format (here .dbf), the file name you want to use and the directory where the exported file will be placed.

Of course, if you want to combine the exported file with another file in the target system, you must use an AREALIST with a geographic key (e.g., the district entity level in winR+) that will match up with the codes in the key for the same areas in your eventual target. The target application will determine whether the source and target files have to use the same keyfield names and whether an index is required to properly link the two keys.

The major problem with this approach is that you may not know whether there will be an acceptable correspondence between the winR+ source file and the target file. The source may have areas not in the target file or vice versa; “inconsistencies” may not be due to substantive errors; e.g., winR+ AREALIST may have the number of persons living on each block in a district while the target may be a map with the same blocks plus some areas without any population —such as traffic islands—. This should be clarified, and any corrective actions taken, prior to making the actual transfer of data.

One of the purposes of the R+ Data Link is exactly this, to determine whether there are inconsistencies and, if so, to detect which records in the source and the target files, respectively, do not match up with counterparts in the other.

If desired, the Link will also append an AREALIST to a target .dbf, for example, with the PAT file of the PCArc/Info or ArcView GIS. The export procedure outlined above simply made the AREALIST available as a .dbf for the target system to import; it did not append the source data to the target file.
Moving an AREALIST to an external target via winR+ Data Link

If you want to send an AREALIST at the district entity level from your Workspace to an external system that uses a .dbf database, as do many GIS, you can use the R+ Data Link with a guided Assist to identify inconsistencies between the AREALIST source and the .dbf of the external system. To illustrate the procedure using an Assist, —and incidentally, to show how the link might be used, for example, when working with very large AREALISTs to see whether there are missing areas or to append them— we have used winR+, itself, to create the target .dbf. The target in the example here is AL_DsmT.dbf, which has 13 districts. The source AREALIST, AL_DsmP, has all but 3 of the same districts; i.e., 3 of them are missing.

Procedure

1. **Open the winR+ Data Link Assist window**: This can be done by clicking on Tools|R+ Data Link or on the icon . It may take a few seconds to appear.

2a **TAB 1. Select Source window** (see Figure 73).

- **Indicate the input file type** by clicking on the circle for winR+ Workspace, since the source is an AREALIST in the Workspace.

  Note in the list of file types that we could have started with a .dbf (.mdb is not implemented in the first release of winR+).

  The first file type, Redatam+ GIS file, allows you to work with a GIS file created in R+, it is not relevant for persons working only with winR+.

- **Select the AREALIST**, here AL_DsmP, in the combo box.

- **Show the File Name for the AREALIST** by clicking the Open button. The File Name appears, which is the Workspace .mdb file with the AREALIST table attached.

- **Indicate the GIS key**, here, district, since that is the AREALIST entity.

- **Click the Next>> button**
Chapter 7. Additional Topics & Procedures — 199

2b **TAB 2. Select Target window** (Figure not shown). This opens because the Next>> button was pressed in the previous step or because Tab was clicked (this window is not shown).

- **Select the Target Type.** Click on the circle for Existing Target file, since the target is a .dbf that we created previously.

- **Define the existing Target File** by clicking the Define existing... button. The Select .dbf file window opens to allow you to find the .dbf file of interest, here AL_DsmT.dbf in the c:\winrplus\work directory.

- **Select the keyfield** in the combo box, in this case, district, since the .dbf was created from an AREALIST at the district level.

- **Click the Next>> button.**

![Figure 73. winR+ Data Link Assist window, TAB 1: Select Source. The source is an AREALIST in the Workspace.](siRGL1.wpg)
2c **TAB 3. Select Fields and Move Data window** (see Figure 75). The list on the left shows the variables first in the source and below them in the list, those in the target (see the *origin* column).

- **Select the Source fields to include** (for the append), which could be all of them. You can select manually by clicking in the far left of a row; the row highlights.

  *The two keyfields (in this case, both District) must not be selected; deselect them if you have used the Select All button(s). Note that the field names and sizes must meet dBASE specifications; you can change the name of any field in the last column.*

- **Select the Source fields to include** in the final output file.

- **Show the inconsistencies, row by row**, by clicking on Add Redflag Control Field to add a field in the output file to indicate whether the Source, the Target or both are present in each row.

- **Check the link**, by clicking on the Check link only button. The fourth window will open or you will receive an error report.
Make the corrections and click the **Check the Link** only button again.

3d **TAB 4. Process Report window** (see Figure 75), which comes up automatically and shows which fields are moved (or will be moved in this case since we used the **Check link only** button in the previous step). At the bottom it indicates inconsistencies, if any; in this case there are “10 matching records; 3 records from target not found in source”. This means that for all others there was a matching source and target key, but not for three areas in the target .dbf.

![Figure 75. winR+ Data Link Assist window, TAB 4: Process Report. The Source AREALIST and Target .dbf have been appended (or it has been simulated by Check link only). The report lists the moved fields and notes 3 records in the Target were not found in the Source file. The inconsistent records can be seen in an external viewer (see text).](siRGL4.wpg)

4. Move the data to append the two files.

**Detecting inconsistencies between the source and target**

4. **Go to any external .dbf viewer to look at the _RedFlag field** such as Excel or QuattroPro, etc. A partial view in Excel shows

<table>
<thead>
<tr>
<th>REDFLAG DISTRICT</th>
<th>_District</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00000 101</td>
<td>El Condado</td>
</tr>
<tr>
<td>1.00000 102</td>
<td>Ciudad Nueva</td>
</tr>
<tr>
<td>1.00000 103</td>
<td>Guadalupe</td>
</tr>
<tr>
<td>1.00000 104</td>
<td>Las Palmas</td>
</tr>
<tr>
<td>2.00000 504</td>
<td>Barrio Verde</td>
</tr>
</tbody>
</table>

The codes for Redflag are:

- 0.0 = record in target but not in source.
1.0 = record in both target and source (consistent)
2.0 = record in source but not in target.

It is then the user’s responsibility to make corrections in the data, which may be due to faulty data or legitimate differences, e.g., areas without population may be traffic islands!

5. Eliminate or accept the inconsistencies.

6. Move the data to append the two files in the Target .dbf. Return to TAB 3 and with the same selection of fields, etc., click the Move data button.

Your data is now ready to continue with further analysis or modeling in a GIS or other external system able to use a .dbf. Depending on the demand from users, later versions of winR+ may include other common formats.
Appendix A.

WHAT IS NEW FOR REDATAM-PLUS USERS

New features in winR+

- **Totally re-written**: winR+ is a new and totally reprogrammed system, for Windows 3.x and Windows 95, that continues and extends the basic ideas and general approach of Redatam-Plus for DOS (abbreviated R+). It is able to do anything that Redatam-Plus can do, and much more.

- **More reliable**: Taking advantage of the Windows and Visual Basic development environment, winR+ is much more reliable than R+. The many components of winR+ are isolated from each other so that improvements (or corrections) in one part do not make problems in others.
- **Uses your existing R+ databases**: Although winR+ is totally rewritten, your existing R+ databases can be used in winR+. The R+ dictionary and related information are quickly converted into a winR+ Workspace; the R+ data files, themselves, do not require any change. Note that the first release of winR+ does not have its own database creation facility so that R+ still must be used for this purpose. An upcoming winR+ revision will have database creation facilities that are much simpler and more integrated than in R+.

**Windows graphical user interface**

winR+ has a full graphical user interface (GUI) with features similar to those found in commercial Windows programs. The mouse-oriented user-interface allows you to open simultaneously various Command Editor and Selection Editor windows, the dictionary, outputs, etc., so that you can have as much information on the screen as you need for your work.

**No inherent memory limitations**

Since winR+ uses all the Windows functionality, including its memory management, there are none of the inherent memory limitations that exist in R+. Of course, computer configurations with more RAM memory will support larger and more numerous outputs.

In addition, a new output, the AREALIST, is available in winR+ for producing tables that are a matrix of geographic areas (rows) with variables (columns) that describe the areas. As an AREALIST is related to the structure of the database, it can be created on the hard disk rather than in the computer memory as required for a crosstabulation. This means that a table with many variables for all the many tens of thousands of city blocks in a very large city can be managed easily in winR+.

**Much faster**

winR+ is much faster than Redatam-Plus. On a given computer and for the same process, winR+ is usually 2 to 4 times faster than R+, depending on the task, although in some cases it can be 10 or 20 times faster.
Open system

R+ was a closed system in that it could only work with its own multisectoral database. winR+ on the other hand, is an open system. That is, it can:

- **Use external files like dbf, mdb**: You can directly access and utilize variables (fields) from your dBASE or Microsoft Access, etc. databases *without* first bringing the variables into your winR+ database—as was necessary in R+. The only special requirement for introducing an external variable into a winR+ command set is that the variable refer to one of the levels in your winR+ database (such as provinces, districts, blocks, houses, households, persons, etc., depending on the structure of your winR+ database).

- **Use multisectoral data without a multisectoral database**: Since the external variables can come from any number of other databases and can be included in a winR+ command set (similar in function to the .spc command file of R+) with a simple command, much of the data in your office, in effect, becomes part of a virtual multisectoral database without any conversion.

In addition, if you frequently use many external dBASE variables, for example, information on each of the external variables and the location of the data can be included in a user's winR+ database dictionary so that the "external" variables become dictionary variables for use in command sets like those in a regular winR+ database. These dBASE variables continue to be part of the .dbf files—the data does not become part of the winR+ database (see the next paragraph)—and therefore can be maintained, updated, corrected, etc., at will with any dBASE or dBASE compatible database software. This frees winR+ from the R+ limitation of working with essentially frozen data (i.e., such as census data that is not expected to be updated or changed), since it is difficult to change existing data within a Redatam database.

- **Save variables directly to your winR+ database**: On the other hand, if you find it convenient, variables that you create within a Command Set or bring in from an external file, can be saved as .bin files, directly to your winR+ database by a simple command. This involves a simple command and is much faster, easier and more convenient than using the relatively complex *Generate* procedure in R+. Furthermore, you
can store the variable data in any directory, not necessarily that where your original database variables are stored.

- **Create external files:** In addition to the usual export of outputs in a format for use in various spreadsheets, etc., with a simple command, you can create outputs directly in the form of dBASE or Microsoft Access files for use in geographical information systems (GIS) and various database managers.

### New command language

`winR+` has a new language, structured in terms of **three** basic commands `RUNDEF`, `DEFINE` and `TABLE`, each of which has a number of possible sub-clauses. The new language is easier to learn and is more intuitive so that it is much simpler to create and read Command Sets than was true with the R+ language. It simplifies hierarchical processing and allows you to use of variables from external databases and to save variables to your `winR+` database as well as to other non-`winR+` formats. It has been found that users of R+ learn the new command language very rapidly. The major disadvantage for them is that their existing programs are **not** usable in `winR+` and no conversion program is planned.

**New type of table, AREALISTS:** `winR+` introduces the AREALIST as a new type of output, in addition to frequencies, crosstabulations and averages. An AREALIST appears at first to be a R+ crosstabulation with individual areas, such as city blocks, in the rows and variables describing the areas in the columns. But as briefly noted in the entry above concerning memory, an AREALIST is managed differently from a crosstabulation in `winR+`—the list can be any length since it uses the hard disk for storage, while the crosstabulation is limited to the computer memory available—and provides a powerful means of linking data with maps and for identifying new Selection Sets based on often complex quantitative criteria. The AREALIST also replaces the WRITE command in Redatam-Plus.
Writing commands with Drag and drop: Assist facilities

Not only is the command language easier to learn and understand, but there are various facilities to help you write commands. A general Assist window gives you all the commands and all the names of the geographic levels and their variables and allows you to drag and drop the commands and variable names into the Command Set that you are writing.

Guided Assists help you to write each of the three commands and their sub-clauses. If you decide to change a command, you can move it back from the Command Set Editor to the Assist for that command and make the changes with the help of the Assist.

There are also expression builder Assists for constructing expressions in commands and for querying AREALISTS.

Tables without writing commands: EASYTABS

While there are many facilities for creating Command Sets to carry out data manipulation and create complex output tables, winR+ also has facilities for visually defining straight-forward frequency, crosstabulation and average tables without writing (or knowing) any commands at all.

Selection of Areas for processing

A key strength of the Redatam software has always been the ability to allow the user to select an area(s) to be processed from a much larger extent (frequently entire countries) and then to process only the data selected. winR+ continues this tradition, but vastly extends the selection capabilities to allow:

- **Easy selection by area codes**: This is the same method used in R+, but with more facilities.

- **Selection by quantitative criteria**: In principle, R+ allowed you to base a selection on simple statistics derived from the database. winR+, on the other hand, allows you to use all the power of its hierarchical data processing to calculate complex indicators upon which to make a selection and then to take advantage of the features of an AREALIST which can be “Queried” as in a database system to make selections of rows (i.e., sub-areas) according to logical expressions. This allows a Selection Set, for example, to be created for the blocks which of all
households have 30% or more with women heads of household who have not completed primary education and have children under 5 years of age. The AREALIST browser also allows manual intervention to select or de-select sub-areas.

- **Selection by map**: If the winR+ map module is available and you have an appropriate Map Definition in your Workspace, you can point and click areas on the map to select them and then automatically create a Selection Set from the map.

**Workspace to keep your work together and safe**

*winR*+ uses the *Workspace* concept to keep all your key “documents” associated with a particular database together in a single file, which is actually a Microsoft Access database. These documents include a copy of the dictionary of the *winR*+ database, your Command Sets and Selection Sets for that database, AREALISTS which you have saved to the Workspace and map definitions, if any. Among the many advantages, this keeps command files, etc., with their associated database (a problem in R+) and allows each person, for example, to have his/her own Workspace on the same computer. Individual documents can be imported or exported from a Workspace. It also facilitates rapid backup of your work since all your Command Sets, etc., are in one file.

**Some graphing and spreadsheet capabilities**

While *winR*+ does not pretend to replace a spreadsheet package for graphing outputs, a variety of graph types are available, including two and three dimensional bar charts, pie charts, line graphs, etc. The AREALIST browser, which is actually a spreadsheet with a row for each area of interest and columns for each variable or field, can be used to carry out manipulations to create new variables and order and select the areas.
Appendix A. What is new for R+ users? — 209

Map display

There is a module available for winR+ that allows you to include map definitions in your Workspace so that you can display results on maps. In the first version of winR+, the map must be in Atlas GIS (.bna) or MapInfo (.mil/.mid) formats, but later versions will support other formats as well. Since winR+ also prepares files for direct export to GIS packages, its map display makes no attempt to provide any of the functionality of a true GIS, but allow rapid spatial display and printing of mapped data.

R+ link to GIS and other database systems

A link is provided to allow winR+ outputs in .dbf and .mdb formats to be related to the equivalent files in database systems, including GIS. When required, the winR+ output can be appended to a GIS attribute table as used in Arc/Info, ArcView, and other GIS packages. The link also provides the user with information on the consistency of the data transfer, namely, whether there are winR+ “areas” that are not on the map, and whether there are map areas for which there are no winR+ data.

Multilingual facilities

winR+ has a multilingual database to facilitate easy translation of the screen and Help text to other languages, in addition to Spanish and English.

Modular structure to facilitate creation of specific applications

winR+ is built of individual program modules —building blocks— that can be used separately or together in other systems. This approach allows reuse of the winR+ components for the construction of other applications for particular purposes. Two such applications that have been written at CELADE are ZonPlan to help identify and localize target populations for social interventions through the mapping of indicators, and a similar but more specific application, INJUMAP, made for the Instituto Nacional de Juventud of Chile (National Youth Institute), with indicators concerning youth in Chile. See also the winR+GIS tools in the next entry.
winR+GIS Spatial Decision Support System tools

winR+ allows you to connect with a set of winR+GIS spatial decision support system tools that utilize sophisticated GIS techniques in easy-to-use packages. As explained in the Acknowledgements and elsewhere in this Manual, winR+ is the product of a joint Project of CELADE with the University of Waterloo of Canada, financed by the International Development Research Centre (IDRC) of Canada and also supported by various other donors. A key purpose of the joint project was to develop spatial decision support systems that use population and other data in connection with winR+ to assist planners and others in a variety of fields and who are not expert in the use of winR+ or GIS, to examine different scenarios so that they make knowledge-based decisions at the local and national levels.

The ZonPlan tool developed at CELADE has already been described in the previous entry of this Appendix. The three winR+GIS tools developed at the Faculty of Environmental Studies, University of Waterloo, Canada are: a) AccessPlan to help improve patient accessibility and clinic resource allocation for primary health care and family planning; b) EduPlan to assist in balancing educational demand and supply, taking into account the spatial distribution of the population and of the educational establishments; and c) TourPlan to aid planners with different perspectives to consider the impact of tourism facility development on the environment and the population, particularly on small islands in the Caribbean. Information on these planning tools is available through CELADE and the University of Waterloo.
Examples of Command Sets in *winR*+ and R+

CmdSet *winR*+ vs. R+. EXAMPLE 1: Recoding, average, crosstab (*winR*+ Tutorial 3b, see Figure 30, page 85)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
</table>
| *01  | RUNDEF T3b_FOR  
    | RUNTITLE "Tut3b: Average age and Education, using DEFINE and FOR"  
    | SELECTION ALL |
| *02  | DEFINE 90person.attstat  
    | AS RECODE 90person.90attend  
    | (0 =0)  
    | (1 =1)  
    | (2-3 =2)  
    | FOR 90person.90age >=5 AND 90person.90age <= 29 |
|      | VALUELABELS  
    | 0 "NoResponse"  
    | 1 "Presently attending"  
    | 2 "Not attending"  
    | VARLABEL "School attendance status"  
    | TYPE INTEGER  
    | RANGE 0 -2 |
| *03  | DEFINE 90person.90age5Rec  
    | AS RECODE 90person.90age  
    | (15-19 =4)  
    | (20-24 =5)  
    | (25-29 =6)  
    | ELSE 0 |

REDATAM-Plus equivalent for Example 1

Selection "c:\redplus\towtmir.sel"  
Universe 90age >= 5 and 90age <= 30  
Areabreak district  
recode 90attend to attstat (0=0)(1=1)(2-3=2)  
Varlabel attstat "school attendance status"  
Valuelabel attstat 0 "no response" 1 "presently attending" 2 "not attending"  
Recode 90age to 90age5re  
(15-19=4)(20-24=5)(25-29=6) else 0  
Varlabel 90age5re "age 5 yr groups"  
Valuelabel 90age5re 4 "15-19" 5 "20-24" 6 "25-29"  
Average 90age by 90edtype by attstat by 90sex  
If 90age5re >= 4 and (90edtype >= 4 and 90edtype <= 9) then begin  
Crosstabs 90age5re by 90attend by 90sex  
Option percent column  
End
CmdSet winR+ vs. R+. EXAMPLE 2:  Create an AREALIST at the block level with aggregate variables at the same level (winR+ Tutorial 4b, see Figure 38 on page 100).

```
*01                     
RUNDEF T4b_ALA
  RUNTITLE "Tutorial 4b: AreaList: Aggregate var at block level"
  SELECTION tutor2b

*02                     
DEFINE 90block.totpers
  AS COUNT 90person
  VARLABEL "Tot pers per block"
  TYPE INTEGER
  RANGE 0-500

*03                     
DEFINE 90block.totfem
  AS COUNT 90person
  FOR 90person.90sex = 2
  VARLABEL "Tot females per block"
  TYPE INTEGER
  RANGE 0-100

*04                     
DEFINE 90block.totmales
  AS COUNT 90person
  FOR 90person.90sex = 1
  VARLABEL "Tot males per block"
  TYPE INTEGER
  RANGE 0-100

*05                     
DEFINE 90block.totHous
  AS COUNT 90housin
  VARLABEL "Tot Houses on block"
  TYPE INTEGER
  RANGE 0-40

*06                     
DEFINE 90block.AvPerpHou
  AS 90block.totpers/90block.totHous
  FOR 90block.totHous <> 0
  VARLABEL "Average Pers/house for blocks"
  TYPE REAL
  OPTIONS
    DEFAULT 99

*07                     
TABLE T4b_ALA
  AS AREALIST
  OF 90block,
    90block.totpers,
    90block.totfem,
    90block.totmales,
    90block.totHous,
    90block.AvPerpHou 2.2
  OUTPUTFILE WORKSPACE T4b_ALA
  OPTIONS
    TITLE "AreaList: Aggregate var at block level"
```

REDATAM-Plus equivalent for Example 2

Selection "c:\redplustutor2a.sel"
Define 90block totpers 0
Define 90block totfem 0
Define 90block totmales 0
Define 90block tothous 0
Define real 90block avperhous 0
Foreach 90block
  Quantify 90person to totpers
  If 90sex=2 then quantify 90person to totfem
  If 90sex=1 then quantify 90person to totmales
  Quantify 90housin to tothous
End
Appendix A. What is new for R+ users? — 213

If tothous <> 0 then compute avperhou=totpers/tothous
Varlabel tothous "average pers/house for blocks"
Write 90block totpers 3.0 totfem 3.0 totmales 3.0 tothous 3.0 avperhou 4.2
Option filename "areaList"

CmdSet winR+ vs. R+, EXAMPLE 3: Determine oldest age in the household (winR+ Figure 44 on page 120).

*01----------------------
RUNDEF oldest_inHH
SELECTION tutor2a

*02----------------------
DEFINE 90housin.oldest
AS 90person.90age
FOR 90housin.oldest<90person.90age
OPTIONS DEFAULT 0

REDATAM-Plus equivalent for Example 3

Selection "c:\redplus\tutor2a.sel"
Define 90housin oldest 0
Foreach 90housin
        If oldest<90age then compute oldest=90age
End
Crosstabs oldest by 90refrig

*03----------------------
TYPE INTEGER
RANGE 0-99

TABLE oldest
AS CROSSTABS
OF 90housin.oldest by 90housin.90refrig
CmdSet winR+ vs. R+. EXAMPLE 4: Determine the sex of the oldest aged person in each household (winR+ Figure 44 on page 120).

```plaintext
*01----------------------
RUNDEF oldestSx
    SELECTION tutor2a
*02----------------------
DEFINE 90housin.oldestAg
    AS 90person.90age
    FOR 90housin.oldestAg < 90person.90age
    VARLABEL "Age of oldest pers in HH"
    OPTIONS
        DEFAULT 0
        TYPE INTEGER
        RANGE 0-99
                      *03----------------------
DEFINE 90housin.oldestSx
    AS 90person.90sex
    FOR 90housin.oldestAg = 90person.90age
    VARLABEL "Sex of oldest pers in HH"
    LIKE 90person.90sex
    TYPE INTEGER
    RANGE 1-2
    OPTIONS
        DEFAULT 0
                      *04----------------------
TABLE oldestsx
    AS CROSSTABS
    OF 90housin.oldestAg by 90housin.oldestSx
```

REDATAM-Plus equivalent for Example 4.
Selection "c:\redplus\towmir.sel"
Define 90housin oldestag 0
Define 90housin oldestsx 0
Foreach 90housin
    If oldestag<90age then begin
        Compute oldestag=90age
        Compute oldestsx=90sex
    End
End
Crosstabs oldestag by oldestsx
Appendix A. What is new for R+ users? — 215

CmdSet winR+ vs. R+. EXAMPLE 5: Determine whether houses are overcrowded (winR+ Figure 44 on page 120).

*01------------------------
RUNDEF OvCrowd
   SELECTION tutor2a
*02-----------------------
DEFINE 90housin.totpers
   AS COUNT 90person
   VARLABEL "No. of persons in HH"
   TYPE INTEGER
   RANGE 0-50
*03-----------------------
DEFINE 90housin.ovcrowding
   AS (90housin.totpers > 3 * 90housin.90slroom)
   VARLABEL "Over-Crowding"

*04-----------------------
VALUELABELS 0 "NO crowding"
   1 "OVER crowding"
   TYPE INTEGER
   RANGE 0-1
   OPTION DEFAULT 0

*04-----------------------
TABLE 1
AS CROSSTABS
OF 90housin.totpers BY 90housin.ovcrowding
   FOR 90housin.totpers > 0 AND 90housin.90soroom > 0

REDATAM-Plus equivalent for Example 5.
Selection "c:\redplus\totwmir.sel"
Define 90housin totpers 0
Define 90housin ovcrowdi 0
   Foreach 90housin
       Quantify 90person to totpers
   End
   Compute ovcrowdi=(totpers>3*90slroom)
   Varlabel ovcrowdi "over-crowding"
   Valuelabel ovcrowdi 0 "no crowding" 1 "over crowding"
   If totpers>0 and 90soroom>0 then begin
   Crosstabs totpers by ovcrowdi
   End
Appendix B.

REFERENCE GUIDE TO THE winR+ COMMAND LANGUAGE

winR+ version 1.1
Overview of the winR+ Command Language

The winR+ command language enables you to speak to the software via Command Sets (CmdSets) that tell winR+ how you want it to process any sub-set of your data defined by your Selection Set (SelSet). The SelSet normally specifies a geographical area, such as a municipality or a city block, within a larger area, such as a city or country. Of course, a SelSet can define the entire database as the area of interest.

The winR+ language is close to natural language and is free-form in that a) any number of blank spaces or lines can be included to facilitate reading; b) commands and clauses of commands can be on more than one line without a continuation character; and c) it is case-insensitive so that upper and lower case characters can be used for improving readability. An asterisk *, placed anywhere before the first character of a line, makes the line into a non-executable comment.

The language is “open” in that it can be used with external files as well as a winR+ database. The external files normally each correspond to a geographical level within a hierarchically-structured winR+ database.

The three basic commands: RUNDEF, DEFINE & TABLE

The winR+ language has only three basic commands:

**RUNDEF** Defines the environment in which a process will occur during a Run, including the mandatory Selection Set (SelSet) that identifies the geographical area to be processed and optional conditions such as the universe to which all processing during the Run will be limited — e.g., only females 15 to 45 —, etc. This is the only required command of a CmdSet.

**DEFINE** Creates new variables, if any, specifies their characteristics, and allows them to be saved, if desired. Various DEFINEs can be used in a single CmdSet.

**TABLE** Describes a specific output and conditions, if any, on its content. Various TABLEs can be used in a single CmdSet.

**Command keywords:** The winR+ command language uses reserved words for the three commands, RUNDEF, DEFINE, and TABLE, and for words that are part of command clauses, such as **FOR, AS,**
RECODE, COUNT, and others to allow you to create take Command Sets. These reserved words are called here command keywords or keywords for short.

**Identifying variables in a Command Set**: Every variable in a winR+ Command Set must always be fully identified by its Entity.Variable name, i.e., its entity and its name separated by a period. Thus, 90person.90sex is the Entity.Variable name of variable 90sex of the entity 90person. In the text, variable name is usually employed in place of Entity.Variable name unless there is ambiguity; the name of the variable without its entity, is called the variable short varlabel (the keyword VARLABEL is always shown in capital letters).

**Command structure and syntax summary**

The three winR+ commands with their respective keyword clauses and syntax are summarized here. The order of the clauses (keywords) within a command is optional, but it is recommended that you use the order approximately as shown to facilitate reading, changing and correcting your CmdSets. A clause can continue onto other lines without a continuation character. However, a new Command starting with one of the three basic commands must begin on a new line.

**Figure 78** lists the three commands with all their possible clauses and keywords available in the first release of winR+.

An example of a typical CmdSet and an outline of its structure is shown in 221. Note that every CmdSet must begin with the RUNDEF command. However, Comments, which have no effect on the execution of a CmdSet, and can be placed anywhere within a CmdSet, can precede the RUNDEF. Any line whose first non-blank character is an asterisk * is treated as a Comment; hence, you can “disconnect” any command or keyword by placing an * before each of its lines.

As noted above, there is no continuation character in winR+. Any clause of any command can continue anywhere on succeeding lines. This allows you to write your Command Sets in a format that facilitates understanding; see the examples under each keyword for suggestions on how you might want to structure your Command Sets.
Operators for expressions in Command Sets

**Arithmetic operators**
- Addition
- Subtraction
- Multiplication
  - Normal division. Gives **TYPE REAL** with decimals
  - Integer division. Gives **TYPE INTEGER**, truncated to lower integer
  - Exponentiation. E.g., $5^2 = 25$
- **MOD** Modulo $y$, gives remainder after dividing by $y$. E.g., $5\text{MOD}2 = 1$
Appendix B. Reference Guide to the Command Language — 221

Relational operators

= Equal to
>= Greater than or equal to
<= Less than or equal to
> Greater than
< Less than
<> Not equal to

Logical operators

AND and
OR or
NOT not

When winR+ evaluates a logical expression, such as \( \text{age} > 10 \) or \( \text{sex} = 1 \), a True result is converted to a 1 and a False result is converted to a 0.

Example of Command Set

| RUNDEF TURC | RUNTITLE "Size of HH by sex of Household Head" |
| Selection list |
| DEFINE 90housin topers |
| AS QUANTITY 90person |
| TYPE INTEGER |
| RANGE 0-3 |
| VARLABEL "Towers in House*" |
| VALUE LABELS |
| 1 * 1-Pers HH* |
| 2 * 2-Pers HH* |
| 3 * 3-Pers HH* |
| 4 * 4-Pers HH* |
| DEFINE 90housin HHHex |
| AS 90person 90inst | |
| FOR 90person 90inst = 1 |
| VARLABEL "Sex of the Household head*" |
| VALUE LABELS |
| 1 * Male HH* |
| 2 * Female HH* |
| RANGE 0-2 |
| TYPE INTEGER |
| OPTIONS DEFAULT 0 |
| TABLE *tab 1 |
| AS CROSSTABS of 90housin topers by 90housin HHHex |
| OPTIONS TITLE "HH size by Sex HH*" |

Structure of Command Set

| RUNDEF command |
| (required & always first command) |
| DEFINE |
| new variables commands |
| (Optional) |
| Any number of variables can be defined |
| TABLE |
| outputs commands |
| (Optional) |
| Any number of outputs can be requested |

Figure 79. An example of a Command Set and the general structure of all Command Sets.
Looking up Keywords in the Reference Guide

Finding a command keyword: The keywords in this Reference are ordered alphabetically within each of the subsections for the three winR+ commands: RUNDEF, DEFINE and TABLE. This is necessary because the individual keywords, which usually initiate clauses within a command, are intimately associated with each of the three Commands of winR+.

The keywords are identified in the left column of these pages as:

COMMAND...KEYWORD

e.g., TABLE...FOR. Hence, if you want information on FOR as used with TABLE, you must look in the left column first for TABLE and then alphabetically for FOR. If you do not know the Command to which a keyword belongs, you will find it quickly by referring to the listing on page 224. The command structure shown on page 219 may also help.

Finally, all individual keywords are listed with their page numbers in the general Index of the Manual.

Command syntax: The syntax is given in this Guide for all the keywords available in the first release of winR+. If you need more information on how to employ a given Keyword, you may find it in the Tutorials and other chapters of this manual by consulting the general index. However, since the Manual treats only the basics of winR+, there are a number of keywords which appear in this Guide, but which are not discussed in the body of the Manual.
Appendix B. Reference Guide to the Command Language — 223

**Typographical conventions for indicating syntax:** A given keyword usually has variables or an expression connected with it and may have various sub-clauses. The following typographical conventions are used to indicate the syntax (these conventions are only to indicate syntax in this manual and are not used in the actual commands and keywords written in a CmdSet):

- **KEYWORD** = Commands & keywords are upper case and bold.
- < > = Variable or other information provided by the user
- <VarID> = entity.variable name
- <expr> = arithmetic expression which may contain variables.
- <TitleString> = title string enclosed within quotation marks: " "
- <zzzID> = Short identification string without quotation marks
- [ ] = Optional keyword or parameter; can be omitted.
- | = Indicates that there are various possibilities, one of which is to be selected. E.g., AS <FREQUENCY|AVERAGE|CROSSTABS|AREALIST> requires that the user include one of these after the AS.
Index to the Command Keywords

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>226</td>
</tr>
<tr>
<td>RUNDEF</td>
<td>227</td>
</tr>
<tr>
<td>RUNDEF...FOR</td>
<td>227</td>
</tr>
<tr>
<td>RUNDEF...RUNTITLE</td>
<td>228</td>
</tr>
<tr>
<td>RUNDEF...SELECTION</td>
<td>228</td>
</tr>
<tr>
<td>RUNDEF...UNIVERSE</td>
<td>229</td>
</tr>
<tr>
<td>DEFINE</td>
<td>230</td>
</tr>
<tr>
<td>DEFINE...AS</td>
<td>231</td>
</tr>
<tr>
<td>DEFINE...AS COUNT</td>
<td>232</td>
</tr>
<tr>
<td>DEFINE...AS DATASET</td>
<td>233</td>
</tr>
<tr>
<td>DEFINE...AS RECODE</td>
<td>236</td>
</tr>
<tr>
<td>DEFINE...AS SUM</td>
<td>238</td>
</tr>
<tr>
<td>DEFINE...AS TOTCASES</td>
<td>239</td>
</tr>
<tr>
<td>DEFINE...FOR</td>
<td>240</td>
</tr>
<tr>
<td>DEFINE...LIKE</td>
<td>241</td>
</tr>
<tr>
<td>DEFINE...OPTIONS</td>
<td>242</td>
</tr>
<tr>
<td>DEFINE...OPTIONS DEFAULT</td>
<td>244</td>
</tr>
<tr>
<td>DEFINE...OPTIONS MISSING</td>
<td>244</td>
</tr>
<tr>
<td>DEFINE...OPTIONS NOTAPPLICABLE</td>
<td>244</td>
</tr>
<tr>
<td>DEFINE...OPTIONS DECIMALS</td>
<td>244</td>
</tr>
<tr>
<td>DEFINE...RANGE</td>
<td>244</td>
</tr>
<tr>
<td>DEFINE...RANGE CODEBOOK</td>
<td>245</td>
</tr>
<tr>
<td>DEFINE...SAVE</td>
<td>245</td>
</tr>
<tr>
<td>DEFINE...TYPE</td>
<td>247</td>
</tr>
<tr>
<td>DEFINE...VALUELABELS</td>
<td>248</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------</td>
</tr>
<tr>
<td>DEFINE...VARIABLE</td>
<td>249</td>
</tr>
<tr>
<td>TABLE</td>
<td>250</td>
</tr>
<tr>
<td>TABLE...AREABREAK</td>
<td>251</td>
</tr>
<tr>
<td>TABLE...AS</td>
<td>252</td>
</tr>
<tr>
<td>TABLE...AS AREALIST OF</td>
<td>252</td>
</tr>
<tr>
<td>TABLE...AS AVERAGE OF</td>
<td>256</td>
</tr>
<tr>
<td>TABLE...AS CROSSTABS OF</td>
<td>258</td>
</tr>
<tr>
<td>TABLE...AS FREQUENCY OF</td>
<td>259</td>
</tr>
<tr>
<td>TABLE...FOR</td>
<td>261</td>
</tr>
<tr>
<td>TABLE...OF</td>
<td>261</td>
</tr>
<tr>
<td>TABLE...OPTIONS</td>
<td>261</td>
</tr>
<tr>
<td>TABLE...OPTIONS MISSING</td>
<td>263</td>
</tr>
<tr>
<td>TABLE...OPTIONS NOKEY</td>
<td>263</td>
</tr>
<tr>
<td>TABLE...OPTIONS OVERWRITE</td>
<td>263</td>
</tr>
<tr>
<td>TABLE...OPTIONS PCR</td>
<td>263</td>
</tr>
<tr>
<td>TABLE...OPTIONS PCT</td>
<td>263</td>
</tr>
<tr>
<td>TABLE...OPTIONS PCC</td>
<td>264</td>
</tr>
<tr>
<td>TABLE...OPTIONS TITLE</td>
<td>264</td>
</tr>
<tr>
<td>TABLE...OPTIONS WEIGHT</td>
<td>264</td>
</tr>
<tr>
<td>TABLE...OPTIONS ZERO</td>
<td>264</td>
</tr>
<tr>
<td>TABLE...OUTPUTFILE</td>
<td>264</td>
</tr>
</tbody>
</table>
COMMENT

- Syntax and use
  *
  < string >
  
  Use: To comment on anything in a Command Set without affecting the execution of the run.

- Examples
  *This is an example of a comment at the beginning of a CmdSet.
  RUNDEF abcd

- Notes
  Any number of comments (remarks) can be placed anywhere in a CmdSet, including before a RUNDEF command. There can be blanks before the asterisk on the line with the comment.
  
  To ignore a keyword during a run of a CmdSet, place an asterisk before the clause; or in front of each of the lines of a command to totally ignore it during a run.

- See also: RUNDEF...RUNTITLE, TABLE...OPTIONS TITLE
RUNDEF syntax and usage

RUNDEF

- Syntax and use

RUNDEF <RunID>

Use: To initiate a CmdSet and provide an identification of the CmdSet. RUNDEF must always be the first line of a new CmdSet.

- Examples

RUNDEF 14c_ALA

- Notes

RUNDEF must always be the first command in a CmdSet. Only a Comment, which is not executable, may precede it.

It is strongly recommended that the <RunID and the name of 8 characters or less with which the CmdSet is Saved in the Workspace, or as a DOS file, should be identical so that printouts and displays can be associated with the originating CmdSet. The RunID must not have blanks.

winR+ has a Guided Assist to help create, edit and check a RUNDEF command with various modifiers; one way of accessing this Assist is by clicking on Tools|RUNDEF Assist on the Main Menu or clicking once on the right mouse button in Windows 3.x and twice in Windows 95. There is also a general Assist with all SelSets, entities, variables and Command Keywords, any item of which can be dragged into the Command Set editor window.

- See also: RUNDEF...RUNTITLE; RUNDEF...SELECTION; RUNDEF...UNIVERSE

RUNDEF...FOR

See RUNDEF...UNIVERSE
RUNDEF...RUNTITLE

■ Syntax and use
RUNTITLE <RunTitleString>  where the <RunTitleString> is enclosed in quotation marks “ “.

Use: To give an overall description of the run.

■ Examples
RUNTITLE “Estimation of poverty by two methods”

■ Notes
It is recommended, but not required, that you use the same phrase for the RUNTITLE and for the CmdSet label in the Workspace.

■ See also: TABLE...OPTIONS...TITLE

RUNDEF...SELECTION

■ Syntax and use
SELECTION <[SELEET] SelSet|SELFILE SelFile | ALL>

Use: To indicate the part of the database to be processed, usually with reference to a geographical area(s). The SELECTION specifies the areas based on the hierarchical structure of the winR+ database.

■ Examples
a) SELECTION tutor2a
b) SELECTION SELSET tutor2a
   Where tutor2a is a Selection Set in the Workspace. Examples a) and b) are equivalent.

c) SELECTION SELFILE c:\winrplus\work\plan.sel
   The path is given to the plan.sel DOS file, which can be a Redatam-Plus selection file or a DOS selection file created in winR+.

d) SELECTION ALL
   The entire database is selected.
Appendix B. Reference Guide to the Command Language — 229

- **Notes**

The **SELECTION** keyword is mandatory in every CmdSet, but it can be used only once per CmdSet.

As indicated in the syntax, **SELECTION** can be used with:

- **SELSET** <SelSet> Selection set saved in the **winR+** Workspace.
- **SELFILE** <SelFile> DOS file. It refers to a traditional REDATAM-Plus selection file.
- **ALL** The entire database is selected.

Since most **winR+** databases are structured in terms of a geographical hierarchy with one or more branches, **SELECTION** normally refers to a geographical area. However, if the database is structured differently, the Selection can be another concept such as date or type of disease, etc., depending on the structure.

- **See also:** TABLE...AREALIST, DEFINE...SAVE

**RUNDEF...UNIVERSE**

- **Syntax and use**

**UNIVERSE** <boolean_expr>

Use: To filter (limit) the cases to be processed and the outputs to be created for the *entire* CmdSet.

- **Examples**

**UNIVERSE 90person.90age >= 5 AND 90person.90age <= 29**

- **Notes**

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*. The boolean expression must be defined only with variables saved permanently on disk; it cannot include variables **DEFINE**d in the current CmdSet.

The keyword **FOR** can be used instead of **UNIVERSE**.

- **See also:** DEFINE...FOR; TABLE...FOR
DEFINE syntax and use

**DEFINE**

- **Syntax and use**

  **DEFINE** <NewVarID> [OVERIDE]

  Use: To create a new variable based on existing variables in a winR+ database and/or external files as well as on previously DEFINEd variables during the same run. Normally the new variable only exists during the current run, but it can be SAVEd for use in future runs.

  **OVERIDE** is a special keyword to change any aspect of a dictionary variable definition during an entire session. When the Workspace is closed or the session ended and then opened again, the variable in the dictionary reverts to its original state.

- **Examples**

  a) **DEFINE** 90housin.totpers

  b) **DEFINE** 90person.90nchild OVERIDE
     LIKE 90person.90nchild
     RANGE 0-30
     OPTIONS NOTAPPLICABLE 99

  This example permits an error in the New Miranda database to be corrected; the database erroneously has *Not Applicable* (99) and *Missing* (98) within the range, which is defined in the dictionary as 0-99. The **OVERIDE** keyword is used here to place the *Missing* and *Not Applicable* categories of **90person.90nchild** outside the range by redefining the variable LIKE the original, but with the range 0-30. Since Not Applicable is 99, all the remaining, code 98, are taken by **winR+** as **Missing**. Note that when the Workspace is closed and then opened again for a new session, the variable reverts to its original state.

- **Notes**

  The new variable must be defined with an *entity.variable name*; the short varlabel is new while the entity name always refers to an existing entity.

  **winR+** has a *Guided Assist* to help create, edit and check a **DEFINE** command with various modifiers; one way of accessing this *Assist* is by
clicking on Tools\RUNDEF Assist on the Main Menu or clicking once on the right mouse button in Windows 3.x and twice in Windows 95. There is also a General Assist with all SelSets, entities, variables and Command Keywords, any item of which can be dragged to the Command Set editor window.

**See also:** RUNDEF; TABLE; DEFINE...AS; DEFINE...FOR; DEFINE...TYPE; DEFINE...RANGE; DEFINE...VARLABEL; DEFINE...VALUETABLES; DEFINE...LIKE; DEFINE...OPTIONS; DEFINE...SAVE

### DEFINE...AS

**Syntax and use**

**AS** <expr>
- COUNT <EntID>
- SUM <VarID>
- RECODE <recode_expr>
- DATASET <DataSetDef>
- TOTCASES (<EntID>)

**Use:** To provide the definition of the new variable through an expression, a hierarchical operation, recode, or use of an external variable.

**Examples**

a) DEFINE 90housin.HHsex
   AS 90person.90sex

b) DEFINE 90housin.parent
   AS 1
   FOR 90person.90relat = 7

c) DEFINE 90housin.90Nitems
   AS 90housin.90waterHH+90housin.90wallsHH+90housin.90roofHH

   Each of the variables 90housin.90waterHH, etc., have been created previously with values of 0 and 1 for *Yes adequate* and *No adequate*, respectively, so that their addition in this example gives the number of adequate items of first necessity in the dwelling unit.

d) DEFINE 90housin.90WallsAdequate
   AS (90housin.90walls < 3 AND 90housin.90walls <> 0)
The expression with AS is a logical expression and is evaluated by winR+ as True or False, which gives the integer values 1 and 0, respectively. Hence, when the logical expression here is True, then the new variable has the value 1; otherwise, it takes the value 0. If there is any doubt about the value of the new variable when the AS expression is False, add the clause OPTIONS DEFAULT 0 (or any other value you wish).

■ Notes

To see the specific syntax and details for each of the keywords associated with AS, look for DEFINE...AS keyword in this Reference Guide.

The new variable is often DEFINEd via the AS clause with a FOR clause to indicate under which conditions the definition applies.

If a logical expression is used, as in example d), it must give a unique result. Normally this will be so when the variables in the expression are at the same entity level or higher than the entity of the DEFINEd variable. If a person level variable, for instance, had been used in the expression of example d), the result would change as each person of the household is processed!

■ See also:  DEFINE...AS COUNT,
            DEFINE...AS SUM,
            DEFINE...AS RECODE,
            DEFINE...AS DATASET,
            DEFINE...FOR

**DEFINE...AS COUNT**

■ Syntax and usage

DEFINE <EntityHigher.Var>
AS COUNT <EntityLower>

Use:  To count the number of elements of a lower entity which are in each of the elements of a higher entity, taking into account only elements in the Selection Set and only those that pass the DEFINE...FOR and/or RUNDEF...UNIVERSE filters, if any.
Appendix B. Reference Guide to the Command Language — 233

■ Examples

a) DEFINE 90housin.TotPers
   AS COUNT 90person

b) DEFINE district.TotMales
   AS COUNT 90person
   FOR 90person.90sex = 1

■ Notes

Since COUNT uses the winR+ hierarchical processing facilities, it must be remembered that winR+ makes a single pass through the data and does everything during that pass. Hence, you can use the result of a COUNT at the district entity level to produce outputs concerning districts; but you cannot use this COUNT result to filter persons since the COUNT will not have completed the count of the persons in the district when a given person is being processed. In special cases, DEFINE...AS TOTALCASES, which also gives a “count”, gets around this restriction.

The built-in default value is 0, that is, if no cases are counted, the result is 0. If there is a FOR filter with the COUNT (or SUM), OPTIONS DEFAULT gives the default value when FOR filters out all cases; if there is no DEFAULT value given, in this situation the result is treated by winR+ as NotApplicable.

■ See also: DEFINE...AS TOTALCASES, DEFINE AS SUM, DEFINE...OPTIONS DEFAULT

DEFINE...AS DATASET

■ Syntax and usage

AS DATASET  < DBF_type>
   MDB*
   WORKSPACE
   ASCII*
   TEXT*
   *
   * Not implemented in the first release of winR+

Use: To enable winR+ to utilize fields (i.e., variables) from an external dataset. The external database may have common formats such as dBASE (.dbf) and MS Access (.mdb), or be an AREALIST from your current Workspace.
Examples

a) DEFINE 90block .NBusin
   AS DATASET DBF3 c:\winrplus\work\lecoact.dbf
   FIELD NStores
   INDEX *
   KEYFIELD ed
   TYPE INTEGER
   RANGE 0-10

The new variable NBusin is defined at the 90block level from a dBASE III database file. The field in the external database is called NStores; it is normally better to use the same name as the original variable, but as here, it is also acceptable to change the name. The KEYFIELD is always the entity level, but uses the name of that field in the external database.

b) DEFINE 90county.TotBirths
   AS DATASET WORKSPACE ExVarAL1
   FIELD TotBirths
   INDEX *
   KEYFIELD County
   TYPE INTEGER
   RANGE 0-1000

The new variable 90county.TotBirths is obtained from the variable (field) TotBirths in the AREALIST ExVarAL1 which is stored in the current Workspace (it is irrelevant whether TotBirths, itself, may have come from the winR+ database or have been brought into the AREALIST from an external file). INDEX * tells winR+ that it must generate an index. Note that the KEYFIELD in the AREALIST is County since, as would be seen by examining the AREALIST (not shown), an AREALIST never uses a FIELD name starting with a number.

Notes

The records of the external database correspond to elements of an entity which may be selectable or non-selectable. Normally, but not necessarily, a KEYFIELD, which corresponds to an entity code, is used as the link to the current winR+ database. A given CmdSet can use only external variables and not refer to any variables in a winR+ database.

If the external source does not correspond fully to the area of the SelSet, of course, only those elements of the entity that do correspond will have values on the external variable.
Appendix B. Reference Guide to the Command Language — 235

The syntax of the specification depends on the type of source database:

1) DBF source files
   Syntax: AS < DBF | DB3 | DB4 | FF2 | FP25 | FP26 > <xBaseFile>
   [ FIELD <string> ]
   [ INDEX < IndexFileName> KEYFIELD <keyname> ]
   [ KEYFIELD <keyname> ]

   DB3, DB4, FP2, FP25 and FP26 are files created in dBASE III,
   dBASE IV, FoxPro v.2.0, FoxPro v.2.5 and FoxPro v.2.5,
   respectively. If in doubt, you can use DBF.

   The FIELD gives the name of the column, i.e., field, of the external
   DBF “table” which is to be taken into winR+ as the DEFINEd
   variable. If it is omitted, the external variable is assumed to be stored
   in the DBF table with the same name as specified in the DEFINE
   command (excluding the entity name). In all cases, the field attribute
   (type and length) in the DBF table must not conflict with the current
   definition in the DEFINE, such as with respect to whether the variable
   is an integer or real variable.

   The KEYFIELD must correspond to the entity to which the new
   variable belongs.

   If an INDEX file name is specified, it must be related to the
   KEYFIELD field; if no index is used, place a space and then an
   asterisk * after INDEX to tell winR+ to generate the index.

   If INDEX * is used and a KEYFIELD is given, the file must to be
   stored in ascending order according to the key. The file is accessed
   sequentially and unmatched recorded are skipped.

   If INDEX * is used and no KEYFIELD is specified, then the dbf file
   is assumed to be "compact" i.e., that the record sequence in the
   external file corresponds one to one with the sequence in the Selection
   Set and, that therefore, the number of records is the same in the
   external and SelSet files. If this is not the case, the results will be
   erroneous and the shortage of data will cause an execution error.
2) **MDB source files** (Microsoft Access .mdb files)

Syntax:  

```
MDB <DatabaseName> <TableName>
[ FIELD <string> ]
[ INDEX <IndexFilename> KEYFIELD <keyname>]
```

<DatabaseName> is the full file name of the Microsoft Access database, such as C:\data\mybase\mymdb.

<tablename> is the table name containing the field within the database.

A real INDEX must be used with a .mdb source file; INDEX * cannot be used. For information on the FIELD, KEYFIELD for MDB source files, refer to DBF source files in this Note.

3) **WORKSPACE source files**

Syntax:  

```
AS WORKSPACE <AREALIST Name>
[ FIELD <string> ]
[ INDEX <indexfilename> KEYFIELD <keyname>]```

The WORKSPACE document refers to an AREALIST located in the current Workspace. The FIELD is the name of the column of interest; the KEYFIELD is the entity level of the AREALIST, i.e., the name of its first column.

- **See also:** TABLE...AREALIST

**DEFINE...AS RECODE**

- **Syntax and use**

```
RECODE [<VarID> | (<arithmetic_expr>) ] <RecodeItem_list>
```

where <RecodeItem_list> has the syntax:

```
[ ( <range_list> = <integer> ) | ELSE <integer> ]
```

Use: To change and/or group the values (categories) of an existing variable by assigning new values.
Appendix B. Reference Guide to the Command Language — 237

Examples

a) DEFINE 90person.90ageGrps
   AS RECODE 90person.90age
   ( 0, 9 = 1)
   (10, 11, 12 TO 29 = 2)
   (30 TO HIGHEST = 3)

b) DEFINE 90person.90ageGrps
   AS RECODE 90person.90age (0-9=1) (10-29=2) ELSE 3

Examples a) and b) give the same results. In the second example, all input values which are not in the two ranges are recoded to 3.

c) DEFINE 90person.agev
   AS RECODE 90person.90age
   VALUE LABELS
   10 "10-14" 11 "15-19" 12 "20-24"
   VARLABEL "Ages of youth"
   RANGE 0-13
   TYPE INTEGER

DEFINE 90person.agev1
   AS 90person.agev
   TYPE INTEGER
   VARLABEL "Ages of Youth; not applic rest!"
   LIKE 90person.agev
   RANGE 0-12
   OPTIONS
   NOTAPPLICABLE 13

The values recoded are those explicitly recoded; all others are “recoded” to their same value. Hence, here ages 0 to 9 will remain single years of age. In the first release of winR+, if the new value 13 is to be treated as NOTAPPLICABLE, as seen in the second DEFINE of this example, another variable, say agev1, must be DEFINEd that is the same as agev, has the RANGE 0-12 and OPTIONS NOTAPPLICABLE 13.

Notes

The existing variable may be a dictionary variable or a variable created with DEFINE in the same run.

The ranges a - b, and a TO b are equivalent. LOWEST and HIGHEST, the minimum and maximum values of the variable, respectively, may be used in specifying a range, thereby avoiding the need to know the actual highest and lowest values.
As seen in example \( d) \) above, if the list of values to be recoded does not include some values, they will be “recoded” to the same value as the original value. Hence,

\[
\text{AS RECODE 90person.90age (30 to HIGHEST=30)}
\]

will recode ages 0 through 29 to themselves and all other ages to 30.

An arithmetic expression in a \text{RECODE} must be enclosed in parenthesis.

- See also: \text{DEFINE...VALUETABLES}

**DEFINE...AS SUM**

- **Syntax and use**

  \[
  \text{DEFINE EntityHigher.Var AS SUM EntityLower.Var}
  \]

  Use: 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 \text{DEFINE...FOR} and/or \text{RUNDEF...UNIVERSE} filters, if any.

- **Examples**

  \[
  \text{DEFINE 90housin.SumAge AS SUM 90person.90age}
  \]

  If there are three persons in a house, aged 25, 35 and 42, the value of \text{90housin.SumAge} for this house will be 102.

- **Notes**

  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 expressed in the Note for \text{DEFINE...AS COUNT} also apply to \text{SUM}.

  Do not use a parenthesis with the variable to be \text{SUMmed}, since it gives a Compile error.
The built-in default value is 0, that is, if no cases are counted, the result is 0. If there is a FOR filter with the SUM (or COUNT), OPTIONS DEFAULT gives the default value when FOR filters out all cases; if there is no DEFAULT value given, in this situation the result is treated by winR+ as NotApplicable.

**See also:** DEFINE...AS COUNT,
DEFINE...OPTIONS DEFAULT

### DEFINE...AS TOTCASES

**Syntax and use**

DEFINE <EntityHigher.Var>
AS TOTCASES(EntityLower)

**Use:** A pre-processing function to provide a “count” of the total number of elements of a lower entity which are in each of the elements of a higher entity, ignoring any FOR or UNIVERSE filters and ignoring the Selection Set boundary. Use with great caution!

**Examples**

a) DEFINE 90housin.TotPersHH
   AS TOTCASES(90person)

b) DEFINE district.TotPersDistr
   AS TOTCASES(90person)

In examples a) and b), the count is for the entire higher entity, 90housin and district, respectively. Since the former is non-selectable, any given element in the Selection Set is always completely in the Selection Set and TOTCASES will give an unambiguous count of each entire housing unit, On the other hand, the district may be cut by the boundary of the Selection Set, but TOTCASES will give the count for the entire Selection Set (COUNT will give the count for the part of each district within the Selection Set).

c) DEFINE 90housin.PersPerBedroom
   AS TOTCASES(90person)/90housin.BedRooms

Since TOTCASES is function it can be used in an arithmetic expression as in this example; it is not necessary to first calculate a variable such as 90housin.TotPersHH in the first example.
Notes

The TOTCASES function, unlike other keywords, can be viewed as making its "count" before beginning the normal hierarchical processing single pass through the data. In fact, the TOTCASES result is not obtained by actually counting cases as does COUNT; rather is obtained from a simple calculation of information stored in the winR+ database telling winR+ where to find the first and last record of a each area. It is for this reason, that FOR, UNIVERSE and Selection Set boundaries are ignored.

It is recommended that you do not use TOTCASES except in special cases when it is the only solution. A solution in many cases when more than one pass is required through the data is to use COUNT to obtain a count that is SAVED in one run and that then is used in the next run.

See also: DEFINE...AS COUNT; DEFINE...SAVE

DEFINE...FOR

Syntax and use

FOR < logical_expression >

Use: Filters the records to include in the new variable DEFINEd, so that only those records for which the logical expression is True take the value given in the AS clause, and all other records are classified in the Not Applicable category.

Examples

a) DEFINE 90housln.LitFemHead
   AS 90person.90liter
   FOR 90person.90relat = 1 AND 90person.90sex = 2

This logical expression defines a household variable as the literacy of the household head when a female; hence, it limits the DEFINEd variable to female household heads. The new variable will be Not Applicable for all other households. Note that this assumes that the census allows only one household head per household; if not, the result will be ambiguous.
Appendix B. Reference Guide to the Command Language — 241

b) DEFINE 90housin.LitFemHead
   AS 90person.90liter
   FOR 90person.90relat = 1 AND 90person.90sex = 2
   OPTIONS DEFAULT 0

   In this second example, the condition is the same as in a), but when it is False, the new variable will take the value indicated in the DEFAULT.

Notes

For a given record being processed, when the FOR expression is True, the new variable takes the value specified in the AS expression or is included in the calculation of the new variable when there is an AS COUNT or AS SUM. When the FOR is False, the new variable takes the value for NOTAPPLICABLE or the OPTION DEFAULT value if it is included; if a COUNT or SUM are being calculated, the record is excluded from the calculation.

See also: DEFINE...OPTIONS DEFAULT;
RUNDEF...UNIVERSE; TABLE...FOR

DEFINE...LIKE

Syntax and use

LIKE < VarID2 >

Use: To copy the attributes of an existing variable to a newly DEFINEd variable.

Examples

a) DEFINE 90housin.90wallsHH
   AS (90housin.walls <= 3 AND 90housin.90walls <> 0)
   VARLABEL "Solid walls"
   LIKE 90housin.90waterHH

   The new variable 90housin.90wallsHH will “inherit” the VALUELABELS, RANGE and TYPE from the variable 90housin.90waterHH defined in a previous command. Since VARLABEL has been specified for the new variable, this is not inherited from the existing variable.
b) DEFINE 90person.90nchild OVERRIDE
LIKE 90person.90nchild
RANGE 0-30
OPTIONS NOTAPPLICABLE 99

This example, permits an error in the New Miranda database to be corrected; the database erroneously has *Not Applicable* (99) and *Missing* (98) within the range, which is defined in the dictionary as 0-99. The **OVERRIDE** keyword is used here to place the *Missing* and *Not Applicable* categories of **90person.90nchild** outside the range by redefining the variable **LIKE** the original, but with the range 0-30. Since *Not Applicable* is 99, all the remaining, code 98, are taken by **winR+** as *Missing*. When the Workspace is closed and then opened again for a new session, the variable reverts to its original state.

**Notes**

The new variable takes on the same attributes as the existing variable except when an attribute is explicitly written in the new variable, such as the case of **VARIABLE** in the above example. The existing variable may be a dictionary or a new variable created previously during the same run.

**See also:**

**DEFINE...OPTIONS**

**Syntax and use**

**OPTIONS**

**DEFAULT** <arithmetic expression>

**MISSING** <constant>

**NOTAPPLICABLE** <constant>

**DECIMALS** <constant>

Use: To assign user-specified values when a given condition exists in a **DEFINE** command.
Appendix B. Reference Guide to the Command Language — 243

Examples

a) OPTIONS
   DEFAULT 0

b) DEFINE 90person.90nchild OVERRIDE
   LIKE 90person.90nchild
   RANGE 0-30
   OPTIONS NOTAPPLICABLE 99

See example b) of DEFINE in the Reference Guide for an explanation.

c) OPTIONS MISSING 98
   OPTIONS NOTAPPLICABLE 99

Notes

The specific OPTIONS are used according to the following rules:

DEFAULT <arithmetic expression>
   Gives the default value of the variable when there is a FOR clause.
   When omitted, the default value of the variable is NotApplicable.
   When there is a COUNT or SUM, it gives the default value if all
   cases are excluded through the FOR; if this OPTION is omitted,
   the built-in default value NotApplicable for COUNT and SUM.

MISSING <constant>
   This option is used when the output variable must be considered
   missing if a missing value appears during a hierarchical process
   such as a COUNT or a SUM.

NOTAPPLICABLE <constant>
   This option is used when the variable must be considered not
   applicable to a given element (record) in an output list such as
   when the element is eliminated with a DEFINE...FOR.

DECIMALS <constant>
   This option specifies the number of decimals for a REAL variable;
   the option is ignored for other types.

See also:  DEFINE...AS FOR; DEFINE...AS COUNT;
           DEFINE...AS SUM; TABLE...OPTIONS
DEFINE...OPTIONS DEFAULT
See DEFINE...OPTIONS

DEFINE...OPTIONS MISSING
See DEFINE...OPTIONS

DEFINE...OPTIONS NOTAPPLICABLE
See DEFINE...OPTIONS

DEFINE...OPTIONS DECIMALS
See DEFINE...OPTIONS

DEFINE...RANGE

- Syntax and use

RANGE [〈range_list〉] [CODEBOOK]

Use: To define the minimum and maximum values that the variable can take or, using CODEBOOK, to find out the range of a variable with a given Command Set and database.

- Examples
a) RANGE 0-17
b) RANGE CODEBOOK
c) RANGE 0-1000 CODEBOOK

As seen in this example, you can use an estimated range, normally with a number far higher than necessary, in order to get initial outputs during the run and also use a CODEBOOK to get the actual minimum and maximum for later runs.

- Notes

Although this is optional for a DEFINE, it should be used in every definition, and must be used if the variable is explicitly employed in TABLE frequency, crosstab and average when categories are used. Knowing the minimum and maximum values of variables to be used in a crosstabulation, frequency or average drastically reduces the processing time.
Appendix B. Reference Guide to the Command Language — 245

*Mis*ssing and Not*Applicable in winR*+ are stored in values that are outside the range. Unless an OPTIONS DEFAULT value is given, the range of a DEFINE with a FOR includes only the values of the new variable when the FOR is True.

- See also: TABLE

DEFINE...RANGE CODEBOOK  See DEFINE...RANGE

DEFINE...SAVE

- Syntax and use

SAVE <dataset specification>

where <dataset specification> is the DOS file name, without any extension, under which the variable will be saved; if no path is given, it will be saved in the current work directory.

Use:  To permanently save the actual values of a new variable so that they can be used in later runs as any other dictionary variable.

- Examples

a)  RUNDEF Save1
   
   SELECTION ALL
   DEFINE 90housin.TotPopHH
   AS COUNT 90person
   VARLABEL "Total persons in household"
   TYPE INTEGER
   RANGE 0-50
   SAVE TotPopHH
b) RUNDEF Save2
   SELECTION ALL
   DEFINE 90housin.NHHHead
   AS COUNT 90person
   FOR 90person.90relat = 1
   OPTIONS DEFAULT 0
   VARLABEL "No. of HHHeads in household"
   RANGE 0-5
   TYPE Integer
   DEFINE 90housin.OneHHH
   AS 1
   FOR 90housin.NHHHead = 1 AND 90housin.TotPopHH > 0
   • Assumes 90housin.TotPopHH was previously saved
   OPTIONS DEFAULT 0
   VARLABEL "Legitimate HH: has 1 & only 1 HHHead"
   VALUE LABELS
   1 "Only 1 HHH"
   0 "0 or 2+ HHH"
   RANGE 0-1
   TYPE INTEGER
   SAVE OneHHH

This example—to create and save a variable identifying households in the database at least one person and with one and only one Household Head—has been given in full to show that the SELECTION must be for the entire database when a SAVE is made and that the SAVED variable should be fully specified. The variable in Example a) is used in b).

Notes

The variable is SAVED in the winR+ internal format, that is, the variable is saved as a separate file with the .bin extension; for this reason, the name cannot be longer than 8 characters and cannot include the entity name. Any directory can be used for the SAVED variable by giving the full path when specifying the file name; if no directory is given, the variable is placed in your current work directory. The dictionary information for the variable is placed in your Workspace. OVERRIDE cannot be used with a SAVE.

The SAVE keyword can be used only in a CmdSet that has a SELECTION that covers the entire database; if a SelSet for less than all the database were used, there would be missing values for the new variables for areas outside the SelSet. SELECTION ALL is provided to facilitate this mandatory selection of the entire database.
Appendix B. Reference Guide to the Command Language — 247

You must specify the **RANGE** of the variable to be saved and should give the **TYPE** and the **VARLABEL** and **VALUELABELS**, if any. If it is **TYPE REAL**, you must use **OPTIONS DECIMAL d**, where \(d\) is the number of digits to the right of the decimals. The **VARLABEL** must be 40 characters or less, which is the limit for the dictionary.

If you use **OPTIONS MISSING** and/or **OPTIONS NOTAPPLICABLE**, their values must be outside the defined range. If neither of these has been specified, \(\text{winR}^+\) will supply them, taking the maximum range + 1 for **NOTAPPLICABLE** and the maximum range + 2 for **MISSING**. Furthermore, if only **MISSING** is specified, **NOTAPPLICABLE** will be assigned the **MISSING** value + 1. Conversely, if only **NOTAPPLICABLE** is specified, **MISSING** will be assigned the **NOTAPPLICABLE** value + 1.

**To delete a SAVEd variable:** Open the Dictionary and click on your SAVEd variable to be eliminated. Click on **Edit|Delete** in the main menu. The CmdSet must be closed first.

- **See also:** **RUNDEF...SELECTION; DEFINE...OPTIONS**

**DEFINE...TYPE**

- **Syntax and use**

  **TYPE [ INTEGER | REAL ]**

  **Use:** To define the type of variable, that is, whether it has **INTEGER** values which have no decimals (e.g., 1; 2; 3; 125, etc.) or it has **REAL** values, which have decimals because the values may not be whole numbers (such as 1.1; 375.67; or 88.0).

- **Examples**

  **TYPE INTEGER**

- **Notes**

  If you do a division of \(\text{Var3} = \text{Var2} / \text{Var1}\), the result is a **TYPE REAL** number. You cannot do a **TABLE...AS FREQUENCY** to see the distribution, which can only be done with **TYPE INTEGER** variables. However, if you want to know the distribution of a **TYPE REAL** variable, say to the first decimal place, **DEFINE** a new variable **AS** the old variable multiplied by 10 and declare it **TYPE INTEGER**. All the
variables will be truncated to the lowest integer value (i.e., 66.6 becomes 66) and a **FREQUENCY** can be obtained.

Note that you can do integer division by using the backslash "\"; the result is truncated to the lower whole integer. Thus, \(4/5 = 0.8\), which is **TYPE REAL**; while \(4\backslash 5 = 0\), which is **TYPE INTEGER**. Similarly, \(14/8 = 1.75\), while \(14\backslash 8 = 1\).

In general, if you can use **TYPE INTEGER** values, your CmdSet will process much more rapidly.

■ **See also:**

**DEFINE...VALUETABLES**

■ **Syntax and use**

```
VALUETABLES < valuelabel_list>
```

To provide descriptive labels for the values of any **TYPE INTEGER** variable.

■ **Examples**

```plaintext
a)
   VALUETABLES 1 "Male" 2 "Female"

b)
   VALUETABLES
      1 "0-4"
      2 "5-9"
```

■ **Notes**

Parentheses **cannot** be used around each label and a comma cannot be used between each label. Therefore, for readability, it is recommended that you organize a **VALUETABLES** as shown in example b).

Since the labels for the values are used in table stubs, the labels should be kept short, particularly for variables used in columns of outputs such as in cross-tabulations.
Appendix B. Reference Guide to the Command Language — 249

- See also: DEFINE...VARLABEL

DEFINE...VARLABEL

- Syntax and use
  VARLABEL < string constant >

  Use: To provide a long descriptive name for a variable.

- Examples
  VARLABEL "5-year age groups"

- Notes
  A VARLABEL can have up to 40 characters.

- See also: DEFINE...VALUETABLES
TABLE syntax and use

TABLE

- Syntax and use

TABLE <TableID>

Use: To produce an output or a related group of outputs which may be one of the following types: FREQUENCY, CROSSTABS, AVERAGE or AREALIST.

- Examples

TABLE Indicators
AS CROSSTABS
OF 90person.vulnerab BY 90person.sex

- Notes

The <TableID> is required. It is used to identify the TABLE in the output report and the display window; hence, although it can consist of more than one word and have as many as 100 characters, it is strongly recommended that it be kept very short, say to 12 characters or less.

Depending on how the AS clause is written, the TABLE command may represent a group of tables (see TABLE...OF).

winR+ has a guided Assist to help create, edit and check a TABLE command with various modifiers; one way of accessing this Assist is by clicking on the right mouse button. There is also a general Assist with all SelSets, entities, variables and Command Keywords, any item of which can be dragged to the Command Set editor window.

- See also: TABLE...AS; TABLE...OF; TABLE...AREABREAK; TABLE...FOR; TABLE...OUTPUTFILE; TABLE...OPTIONS
TABLE...AREABREAK

- Syntax and use

AREABREAK <entity>

Use: To produce the same TABLE output for each element of a given entity, e.g., a district, within the Selection Set plus a summary TABLE for the entire Selection Set.

- Examples

RUNDEF
SELECTION tutor2a
...
...
TABLE Indicators
AS CROSSTABS
OF 90person.vulnerab BY 90person.90sex
AREABREAK district

The AREABREAK in this example will produce the same crosstabulation for each district within, or part of, the SELECTION area and a summary crosstabulation for the entire SELECTION.

- Notes

An AREABREAK affects only the specific table within which it appears. The summary TABLE for an AREABREAK is the same as that which would have been obtained without the AREABREAK. Consequently, in the above example, if a given element, say district of the district entity is not entirely within the Selection, only the part of district within it is reported in the output TABLE.

The entity indicated in an AREABREAK must be selectable; thus, the house, household and person entities often used in a census database cannot be used in an AREABREAK, since they are usually unselectable; all higher entities normally are selectable.

Note that AREABREAK cannot be used when more than one table is created in a given FREQUENCY, CROSSTABS or AVERAGE.

- See also: RUNDEF...SELECTION; TABLE...AREALIST
TABLE...AS

- Syntax and use

AS FREQUENCY
AVERAGE
CROSSTABS
AREALIST

Use: To indicate the type of tabular output requested for a specific TABLE command.

- Examples

a) TABLE freq1
   AS FREQUENCY

b) TABLE AL1
   AS AREALIST

- Notes

winR+ also has the EasyTabs feature that permits the creation of frequencies, averages and crosstabulations without writing a Command Set. See the Tutorials of the Basics of winR+ Manual or the winR+ Help.

winR+ has a guided Assist to help create and edit each of the types of TABLE and their various modifiers; one way of accessing this Assist is by clicking on the right mouse button. There is also a general Assist with all SelSets, entities, variables and Command Keywords, any item of which can be dragged to the Command Set editor window.

- See also: TABLE...AS AREALIST OF; TABLE...AS AVERAGE OF; TABLE...AS CROSSTABS OF;
TABLE...AS FREQUENCY OF

TABLE...AS AREALIST OF

- Syntax and use

AS AREALIST
   OF <KeyEntity> [,<VarID> <alias> <fmt> [,<VarID> <alias> <fmt>]
   [...]]

where:
Appendix B. Reference Guide to the Command Language

<KeyEntity> Indicates the entity which will be used to define the rows of the AREALIST.

<VarID> Is the Entity.Variable name for each variable to be included as a column in the AREALIST output.

<alias> Is an optional short name for the Entity.Variable name that will be used as a column title for the variable in the AREALIST output (see the Notes).

<fmt x,y> Is the optional format in which the output of the variable will appear in the AREALIST output, where x is the number of characters to the left of the decimal and y the number to the right.

Use: To produce a tabular output—see the Figure—in which the rows refer to elements of an entity and the columns to variables (also called fields); an AREALIST is often employed as an input to a map display showing the areal distribution of a given variable over elements of a given entity, e.g., districts, where each district is colored according to its value of the variable.

![AREALIST](image.png)

Figure 80. A winR+ window showing an AREALIST for the county entity of all of New Miranda. The rows are each a county and the columns are each variables (also called fields).

Examples

a) TABLE ALst1
   AS AREALIST
   OF 90block,
   90block.totpers,
90block.AvPersPerHouse AvPpH 3.2
OUTPUTFILE WORKSPACE ALst1

The <KeyEntity> in this example is 90block, so the rows of the AREALIST will be the blocks within the Selection area. The alias AvPpH will be used as the short column title for the 90block. AvPersPerHouse variable and the format of this variable will be 3.2, that is, three digits to the left of the decimal and two digits to the right of the decimal. The AREALIST will be stored in the current Workspace under the name ALst1.

b) Table ALst2
   AS AREALIST
   OF 90block,
       90block.totpers,
       90person.90sex sex,
       district.rainfall
   OUTPUTFILE DBF c:\plandir\ALst2
   OPTIONS OVERWRITE

This is similar to example a) except for the following: The output AREALIST has been sent as a DOS file in dBASE III format to the directory c:\plan where it is stored with a .dbf extension. If no path were given, the current work directory would be used. Note that this AREALIST has variables from a lower, the same and a higher entity than 90block. 90sex must be given an alias, sex here, since a column in an AREALIST must not begin with a number. If there is an existing DBF with the same name, OVERWRITE will allow the new file to be written (see the Notes below).

= Notes

The first two variables on the OF list in example b), each have one value for each block since each variable is at the 90block entity level (see the figure). The next to last variable, 90person.90sex, is at a lower level and, therefore, has no unique value for each block; in such cases, winR+ gives the distribution over the range of the variable, i.e., in this case, there will be two columns: Sex1 and Sex2, the VariableCodes for males and females, respectively, and the number of cases in each category will be given for each of the elements of the KeyEntity, here 90block. The lower entity level variables must have a RANGE specified.

Variables from higher entities than the KeyEntity must be dictionary variables (which may include those created and SAVE'd during an earlier
run). The higher entity variables cannot be created using COUNT or other hierarchical process in the same run as the output AREALIST.

An Alias for a variable is optional except in 3 mandatory cases: 1) When the OF list has two short varlabels with the same name; another name via an alias must be given to at least one short varlabel to avoid having two columns of the AREALIST with the same names; 2) When a variable has a short varlabel that begins with a number, e.g., 90sex, an alias must be given that begins with a letter; 3) When the variable varlabel has more than 10 characters, including if the code is added for a lower entity variable, the digits of the code; e.g., in a case when there are two digits, the alias can be no more than 8 characters.

The codes of the entity elements listed in an AREALIST are given unless NOKEY is used in the OPTIONS. The names of the elements, if any, can be included as a column variable; when an entity has element names, the variable is found in the dictionary as @EntityName, e.g., @distri.

If the OUTPUTFILE exists, winR+ will tell you during compilation and you can decided whether to erase the existing AREALIST or not. If you want to overwrite an existing external AREALIST, i.e., one that is not in the Workspace, you also can add OPTIONS OVERWRITE. Since this does not apply to your Workspace; if you get an error message that the AREALIST exists in your Workspace, you must erase it, or change the name of the new one in the CmdSet.

The OUTPUTFILE types for an AREALIST are:

**WORKSPACE**  Stored in your current Workspace as a document

**ASCII**  Stored in a formatted DOS file with extension .dat. If no path is specified, the file is placed in your current work directory. [Not available in the first release of winR+]

**DBF**  Stored in dBASE III file with extension dbf. If no path is specified, the file is placed in your current work directory.

**MDB**  Stored in a Microsoft Access MDB file with extension mdb. If no path is specified, the file is placed in your current work directory. [Not available in the first release of winR+]
**SELECTION**

The AREALIST output is converted to a Selection Set and stored in the current Workspace as a Selection document.

See TABLE...OUTPUTFILE for more information on each type.

It is important to realize that an AREALIST is different from the other types of TABLE, even though it may seem similar to a CROSSTABS. Unlike a CROSSTABS, which is stored in memory during processing since it is not known which cells will be increased until the last record of the run is processed, each row (i.e., area) of an AREALIST is completed before the next is started, allowing the AREALIST to be built-up on the hard disk. This means that the many tens of thousands of blocks of a large city can be included in an AREALIST with many variables, while such crosstabulation would soon run out of memory.

*See also:* TABLE...OPTIONS; TABLE...OUTPUTFILE;
TABLE...FOR; TABLE...AREABREAK;
TABLE...OF; RUNDEF...SELECTION; TABLE...AS FREQUENCY OF; TABLE...AS CROSSTABS OF;
OPTIONS OVERWRITE

### TABLE...AS AVERAGE OF

**Syntax and use**

AS AVERAGE

OF <varID_list1> [BY <varID_list2> [BY <varID_list3>
    [BY <varID_list4>]]]

---

*Use:* To produce an average of the first variable(s) with up to three additional variables as controls.

**Examples**

a) **TABLE Av1**

    AS AVERAGE
    OF 90person.90age BY 90person.90sex

b) **TABLE Av2**

    AS AVERAGE
    OF 90person.90age, 90person.90YrsEduc
    BY 90person.90sex, 90housin.90water
Example b) will produce an average of 90age for males and females, just like a), but in the second example, there will also be an average for 90YrsEduc for each sex. In addition, there will be a second set of the two averages by whether the persons live in a house with or without water.

Notes

The average is always calculated with the variable(s) in the list before the first BY. Of course, they should normally be quantitative variables, like age, to get a meaningful average. Care must be taken to exclude non-meaningful values from the average using FOR, such as when No response is coded as 99 for the variable Number of Children-ever-born. The variables after the first BY must each have a RANGE defined.

AREABREAK cannot be used when more than one output table is produced for AVERAGE (or any other type of TABLE).

The output will be displayed on the screen and can be printed, but is not saved from within the Command Set unless OUTPUTFILE is used. The OUTPUTFILE types available for an AVERAGE (or CROSSTABS or FREQUENCY) are:

- **TEXT**
  
  Stored as a DOS file with a .txt extension in the path specified. If no path is given, the current work directory will be used. [Not available in the first public release of winR+.]

- **SPREADSHEET**
  
  Stored as an Excel file with an .xls extension in the path specified. If no path is given, the current work directory will be used. [Not available in the first public release of winR+.]

*winR+* has a guided *Assist* to help create, edit and check a TABLE...AS AVERAGE command with various modifiers; one way of accessing this *Assist* is by clicking on the right mouse button. There is also a general *Assist* with all SelSets, entities, variables and Command Keywords, any item of which can be dragged to the Command Set editor window.

See TABLE...OUTPUTFILE for more information on each type.
See also:  TABLE...OPTIONS; TABLE...OUTPUTFILE; TABLE...FOR; TABLE...AS FREQUENCY OF; TABLE...AS CROSSTABS OF; TABLE...AS AREALIST OF; TABLE...AREABREAK

TABLE...AS CROSSTABS OF

■ Syntax and use

AS CROSSTABS
  OF <varID_list1> BY <varID_list2> [BY <varID_list3> ]

Use: To produce a crosstabulation with the first variable(s) as the row, the second variable(s) as the column and the optional third variable(s) as the control ("panel") variable(s).

■ Examples

a) TABLE Xtabs1
   AS CROSSTABS
     OF 90person.90relat BY 90person.90sex BY 90person.90AgeGrp5

   Three-way crosstabulation where 90relat is the row variable, 90sex is the column variable and 90AgeGrp5 (5-year age groups) is the panel variable, i.e., for each category of the panel variable, there is a crosstabulation of the other two variables.

b) TABLE Xtabs2
   AS CROSSTABS
     OF 90housin.90water, 90housin.90light
     BY 90person.90sex

   Two separate crosstabs: 1) 90water by 90sex and 2) 90light by 90sex will be produced. Note that unlike in example a), this example is a crosstabulation at two different entity levels, in which case, the tabulation is done in terms of the lower entity, here 90person; hence, for example, the first table will tell you how many males and how many females have water and how many do not.

■ Notes

The RANGE of each variable in a CROSSTAB must be defined.

The code and/or name, if any, of an entity element cannot be used as a row or column in a CROSSTABS; to get such a listing of these with variables of interest, use AREALIST.
Appendix B. Reference Guide to the Command Language — 259

AREABREAK cannot be used when more than one output table is produced for CROSSTABS (or any other type of TABLE).

The output will be displayed on the screen and can be printed, but is not saved from within the Command Set unless OUTPUTFILE is used. The OUTPUTFILE types available for an CROSSTABS (or AVERAGE or FREQUENCY) are:

**TEXT**

Stored as a DOS file with an .txt extension in the path specified. If no path is given, the current work directory will be used. [Not available in the first public release of winR+.]

**SPREADSHEET**

Stored as a Excel file with an .xls extension in the path specified. If no path is given, the current work directory will be used. [Not available in the first public release of winR+.]

**winR+** has a *Guided Assist* to help create, edit and check a TABLE...AS CROSSTABS command with various modifiers; one way of accessing this Assist is by clicking on the right mouse button (once in Windows 3.x and twice in Windows 95). There is also a *General Assist* with all SelSets, entities, variables and Command Keywords, any item of which can be dragged to the Command Set editor window.

- See also: TABLE...OPTIONS; TABLE...OUTPUTFILE; TABLE...FOR; TABLE...AS FREQUENCY OF; TABLE...AS AVERAGE OF; TABLE...AS AREALIST OF; TABLE...AREABREAK

**TABLE...AS FREQUENCY OF**

- **Syntax and use**

**AS FREQUENCY**

**OF** <VarID_list>

Use: To produce a frequency distribution ("marginals") of a variable or of each variable on a list.
Examples

a) TABLE Freq1
   AS FREQUENCY
   OF 90person.90relat

b) TABLE Freq2
   AS FREQUENCY
   OF 90person.90relat. 90person.AgeGrp5, 90housin.totpers

A separate frequency distribution will be produced for each of these variables; note that the list can have variables at different entity levels.

Notes

The codes and names, if any, of entity elements cannot be tabulated in a FREQUENCY; to get such a listing, use AREALIST. The variable to be tabulated must have a RANGE defined.

The output will be displayed on the screen and can be printed, but is not saved from within the Command Set unless OUTPUTFILE is used. The OUTPUTFILE types available for a FREQUENCY (or CROSSTABS or AVERAGE) are:

- TEXT
  Stored as a DOS file with an .txt extension in the path specified. If no path is given, the current work directory will be used. [Not available in the first public release of winR+.

- SPREADSHEET
  Stored as an Excel file with an .xls extension in the path specified. If no path is given, the current work directory will be used. [Not available in the first public release of winR+.

winR+ has a Guided Assist to help create, edit and check a TABLE...AS FREQUENCY command with various modifiers; one way of accessing this Assist is by clicking on the right mouse button (once in Windows 3.x and twice in Windows 95). There is also a General Assist with all SelSets, entities, variables and Command Keywords, any item of which can be dragged to the Command Set editor window.

See also: TABLE...OPTIONS; TABLE...OUTPUTFILE; TABLE...FOR; TABLE...AS CROSSTABS OF; TABLE...AS AVERAGE OF; TABLE...AS AREALIST OF; TABLE...AREABREAK
Appendix B. Reference Guide to the Command Language — 261

TABLE...FOR

- Syntax and use
  
  FOR <logical_expression>
  
  Use: Filters the records to be used in the TABLE, so that only those are included for which the logical expression is True.

- Examples
  
  a) TABLE Education
     AS CROSSTABS
     OF 90person.90AgeGrp5 BY 90person.90attend BY 90person.90sex
     FOR 90person.90AgeGrp5 >= 4 AND (90person.90edtype >=4 AND 90person.90edtype <=9)
  
  b) {same first three lines of example a} }
     FOR 90person.90sex = 2 AND 90housin.90water
  
  Note that in example b), the FOR involves variables at different entity levels; the higher entity level variable must be a dictionary variable and cannot be one calculated by hierarchical processing in the CmdSet.

- Notes
  
  The records included in a given output are determined not only by a TABLE...FOR, but also by the RUNDEF...UNIVERSE filter, if any, affecting the entire run, as well as the variables used in the TABLE that may have been created with DEFINE...FOR.

  A FOR must apply to the entity level of an AREALIST.

- See also: RUNDEF...UNIVERSE; DEFINE...FOR

TABLE...OF

See TABLE...AS AREALIST OF
See TABLE...AS AVERAGE OF
See TABLE...AS CROSSTABS OF
See TABLE...AS FREQUENCY OF

TABLE...OPTIONS

- Syntax and use
OPTIONS
  MISSING
  NOKEY
  OVERWRITE
  PCR PCC PCT
  TITLE <table_title>
  WEIGHT <VarID>
  ZERO

Use: To assign user-specified options to a TABLE; the options that can be used depend on the type of TABLE specified in the AS clause.

- Examples

  a) TABLE Xtabs1
     AS CROSSTABS
     ...
     OPTIONS
     PCR PCT
  b) TABLE Xtabs2
     AS CROSSTABS
     ...
     OPTIONS
     MISSING
     TITLE "Household vulnerability BY number items in house"
  c) TABLE F2
     AS FREQUENCY
     OF 90person.90AgeGrp5
     FOR 90person.90nchild < 30 AND 90person.90sex = 2
     OPTIONS
     WEIGHT 90person.90nchild
     TITLE "Children ever born alive woman’s age"

Calculation of the number of Children Ever Born (CEB) to women in 5-year age groups. Note that the FOR limits the variable 90person.90nchild to less than 30, since this variable has the values 98 and 99 for missing and NotApplicable.

- Notes

Specific OPTIONS can be used, or not, according to the following rules:

MISSING

Missing values will be included in the output TABLE, i.e., the default is to exclude missing values from the table.
Appendix B. Reference Guide to the Command Language — 263

NOKEY
Used with an AREALIST to omit the first column which normally has the area code.

OVERWRITE
Used when creating an external AREALIST, i.e., a .dbf, etc., outside the Workspace; an existing file of the same name will be overwritten without warning. It has no effect if the AREALIST is being sent to the Workspace; if an AREALIST of the same name exists, a compile error will be reported and the document must be renamed or erased prior to running the CmdSet again.

PCR PCC PCT
Calculate the Row, Column and Total percentages for an AVERAGE or a CROSSTABS. One or more of these can be used; do not separate by commas.

TITLE <table_title>
Optional TABLE title enclosed in quotation marks.

WEIGHT <expression>
Weights each case in the tabulation process. The WEIGHTing expression, which can have one or more variables and constants, applies only to the current TABLE. If no WEIGHT is used, in effect, the weight is 1.

ZERO
Rows with all zeros and Columns with all zeros are included in the output TABLE, i.e., the default is to exclude zero rows and columns.

See also:  DEFINE...OPTIONS; RUNDEF...RUNTITLE

TABLE...OPTIONS MISSING
TABLE...OPTIONS NOKEY
TABLE...OPTIONS OVERWRITE
TABLE...OPTIONS PCR
TABLE...OPTIONS PCT
See TABLE...OPTIONS
TABLE...OPTIONS PCC
See TABLE...OPTIONS
TABLE...OPTIONS TITLE
See TABLE...OPTIONS
TABLE...OPTIONS WEIGHT
See TABLE...OPTIONS
TABLE...OPTIONS ZERO
See TABLE...OPTIONS

TABLE...OUTPUTFILE

Syntax and use

OUTPUTFILE <FileType> <FileSpecifications>

Use: To specify a file format for storing an output TABLE; this is required for an AREALIST, but is optional for a FREQUENCY, CROSSTABS and AVERAGE. The formats and specifications depend on the type of TABLE specified in the AS clause.

Examples

a) TABLE abc
   AS AREALIST
   OF 90block,
   90block.totpers
   OUTPUTFILE WORKSPACE abc

   The AREALIST is sent to the Workspace, i.e., is FileType = Workspace, and has the name abc. It can be viewed from the Workspace. If abc already exists in the Workspace, you must erase it or change the name (OPTIONS OVERWRITE can be used to replace an existing AREALIST only when it is an external file such as a DBF).

b) {same first three lines of example a}
   OUTPUTFILE DBF c:\scratch1\abc

   The AREALIST, with the name abc and the extension.dbf is sent as a DBF file to the directory c:\\scratch1
Appendix B. Reference Guide to the Command Language  —  265

Notes

As indicated in the Use Section of this entry, the OUTPUTFILEs type and specification varies with the type of TABLE. The outline that follows will help to clarify when and how to use this the OUTPUTFILE with different types of TABLE. Refer to entries of each type for more detail.

If an AREALIST already exists as a dbf or other external file, you will receive a warning during compilation; if you want to continue, you can Delete the AREALIST or add OPTIONS OVERWRITE so that the the AREALIST is overwritten when the CmdSet is run. Otherwise, change the name of the new AREALIST. If the AREALIST is in your Workspace, you must erase it since OVERWRITE has no effect.

<table>
<thead>
<tr>
<th>OUTPUTFILE &lt;FileType&gt;</th>
<th>TABLE output type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FREQUENCY</td>
</tr>
<tr>
<td>None</td>
<td>Display/Print</td>
</tr>
<tr>
<td>TEXT*</td>
<td>filename.txt</td>
</tr>
<tr>
<td>SPREADSHEET*</td>
<td>filename.xls</td>
</tr>
<tr>
<td>WORKSPACE</td>
<td>---</td>
</tr>
<tr>
<td>ASCII</td>
<td>---</td>
</tr>
<tr>
<td>DBF (dBASE III)</td>
<td>---</td>
</tr>
<tr>
<td>MDB (MS Access)*</td>
<td>---</td>
</tr>
<tr>
<td>SELECTION</td>
<td>---</td>
</tr>
</tbody>
</table>

* Not available in the first release of winR+. ¹ An AREALIST with OUTPUTFILE SELECTION generates a normal Selection Set which is stored as a document in the current Workspace.

The OUTPUTFILE SPREADSHEET is an Excel file with a .xls extension; it can be imported into Lotus and Quattro Pro.
See also: TABLE...AS CROSSTABS OF; TABLE...AS AVERAGE OF; TABLE...AS FREQUENCY OF; TABLE...AS AREALIST OF; TABLE...AREABREAK
Appendix C.

GLOSSARY

If you do not find the word(s) that you are looking for in this Glossary, please check the General Index of the Manual.

algorithm  Procedure or set of steps to follow to attain a desired result. The Command Set exemplifies the algorithm.

active window  In the Windows operating system, if more than one window is open, the window you are currently working with is the active window. You can make a window active by clicking on its title bar.

AREALIST  An AREALIST is one of the four standard outputs of winR+; the others are crosstabulations, frequencies and averages. An AREALIST has the appearance of a spreadsheet in which the rows are a list of elements of an entity, e.g., districts, and the columns are variables describing each element, e.g., the number of houses in the district. AREALISTS are used by winR+ to send information to maps, to select items by query, create Selection Sets based on quantitative data, etc.

Assist windows  Assist windows help you to work within winR+. They are similar to what some other software call Wizards.
268 — Basics of winR+

_Arc/Info®_ Geographical Information System (GIS) developed by ESRI, Inc.

_ArcView®_ Map viewing system of ESRI, Inc, with some Geographical Information System (GIS) capabilities. Used with _Arc/Info_.

ASCII Standard text DOS characters and punctuation marks on the screen and United States keyboards have a number associated with them according to the _American Standard Code for Information Interchange_ (ASCII). Spanish accented letters and some other non-English characters are also included in the extended ASCII characters. There is a code between 0 and 255 for each character.

_AtlasGIS®_ Geographic Information System. _winR_+ can work with map files produced in _AtlasGIS_.

_.bna_ The extension of an _AtlasGIS®_ map file; _winR_+ can display a variable from an AREALIST on a map in _bna_ format. _winR_+ can store a Map Definitions with this format in your Workspace.

CELADE Latin American Demographic Center (Centro Latinoamericano de Demografía), Casilla 91, Santiago, Chile.

categories Qualitative variables have “values” that are words like _male_ and _female_, or _Yes_ and _No_, etc. For data processing, these categories are each assigned a number, e.g., _male_=1 and _female_=2. See also _classes_.

classes The values of a numerical variable, like a percentage or the number of males can be divided into a number of _classes_, which are groups of values. Each class is defined by a minimum and maximum value. The classes, also called _categories_, may be given names like Low, Medium and High, and assigned integer values, e.g., _Low_=1, _Medium_=2 and _High_=3.

click Place your cursor over the button or item of interest on the screen and press (_click_ once) the left button of the mouse.

CIDA Canadian International Development Agency.

CmdSet Abbreviation of _Command Set_.

columns In the usual dBASE table and in _AREALISTS_, the _fields_ or _variables_ are the columns —the _records_ are the _rows_—.
Appendix C. Glossary — 269

**combo box**
A text box with an arrow on the right that when clicked upon, opens a list. Clicking on an item selects the item that will appear in the text box when the list closes. E.g.,

![ComboBox](image)

**default**
If no instruction concerning a particular action is given by the user of winR+, the software normally has a “built-in” decision that is taken by default so that the system can proceed. For example, if a directory is not indicated for storing a file that you create, it will be placed in your work directory, which is your default directory.

**dBASE®**
The database file format that uses the .dbf extension. Sometimes called xBASE since there are many “dialects” of the dBASE format.

**.dbf**
The file extension of a dBASE file, that is, .dbf.

**directory**
Directories and sub-directories are named areas on a computer hard disk where files are stored. Also called a “folder”.

**DOS**
The Disk Operating System used by IBM PC compatible microcomputers.

**ECLAC**
Economic Commission for Latin America and the Caribbean. In Spanish: CEPAL.

**element**
An element is one item of an entity, e.g., the province of Atacama of the entity province. See “entity”.

**entity**
An entity is a level within the structure of the hierarchical winR+ database. As most winR+ databases are geographically structured, the entities are normally levels such as provinces, counties, districts, city blocks, houses, households and persons. Each such entity has individual elements such as the provinces of San Antonio, Concepción, Puerto Nuevo, etc.

**Entity.Variable**
The full name of a variable that must be used in a winR+ Command Set. The entity name and the short variable name (also called here a short varlabel) of a variable are written with a period between the two, e.g., 90person.90sex, where 90person the entity of the variable and 90sex is the short varlabel, respectively.

**Excel®**
Spreadsheet package of Microsoft Corp.
external variables

Variables that winR+ can assess in external database files with the .dbf format — and in other formats in later winR+ versions —. The external database must have records that refer to an entity in the winR+ database that you are using.

fields

In dBASE (.dbf) and other commercial database systems, the variables are usually called fields. Both terms are used with AREALISTS in winR+. In dBASE tables and in AREALISTS, the fields or variables are the columns, and the records are the rows.

folder

See “Directory”

GIS

Geographic Information System

IDRC

International Development Research Centre, Canada. Financed the various Redatam-related projects.

keywords

See “Command keywords”.

Lotus 1-2-3®

Spreadsheet package of Lotus Development Corp.

Main Menu

Bar at the top of the screen on the line after the program name. Contains items such as: File Workspace Tools Windows Help. The items vary depending on the windows open, etc. Also call the “Main Menu bar”.

map definition

The map definition is one of the four types of documents that can be stored in a Workspace. The map definition makes it possible for winR+ to connect an AREALIST to a map display. In the first version of winR+, only digital maps with the .bna format of Atlas GIS or the format of .mif/.mid of MapInfo can be used.

MapInfo®

Geographic Information System. winR+ can work with maps produced by system.

.mif/.mid

The extensions of an MapInfo® map file; winR+ can display a variable from an AREALIST on a map in this format. winR+ can store a Map Definitions with this format in your Workspace.

.mdb

Extension used with a Microsoft Access database.

Microsoft Access®

The database file format with the .mdb extension; the database management system package of Microsoft Corp. that uses the .mdb format.

Quattro Pro®

Spreadsheet package, originally by Borland, Inc., now of Corel, Inc.
Appendix C. Glossary — 271

R+  Redatam-Plus (REtrieval of DATa for small Areas by Microcomputer). The second generation of the Redatam package.

Redatam  REtrieval of DATa for small Areas by Microcomputer. The first generation of the Redatam package; for DOS. Redatam 3.x was the last and most widespread version.

Redatam-Plus©  REtrieval of DATa for small Areas by Microcomputer. Abbreviated R+.

rows  In dBASE tables and in AREALISTS, the fields or variables are the columns, and the records are the rows.

run  All that winR+ does while carrying out a single Command Set. Sometimes called a job.

Selection Set  Identifies the cases to be processed during a winR+ run, normally by specifying a geographical area. Abbreviated SelSet.

SelSet  An abbreviation of Selection Set.

short varlabel  The name of a variable excluding its entity. Note that the command keyword VARLABEL is used to add a descriptive long name to a variable.

.spc  File extension of a command file in Redatam-Plus (DOS). In winR+ a command set is stored in the Workspace and, therefore, the command set does not have a file extension.

title bar  The uppermost bar of a window in Windows.

UWaterloo  University of Waterloo, Ontario, Canada. The work on the joint CELADE-UWaterloo project, within which winR+ and various R+GIS tools were developed, was carried out with the Faculty of Environmental Studies of the University of Waterloo.

variables  Information about individual elements of an entity is stored in variables. For example, age, sex, occupation, etc., are variables of the person entity. In an AREALIST or a .dbf file, the variables are the columns. In an AREALIST or a .dbf, variables are also called fields.

winR+©  Redatam-Plus for Windows. The third and latest generation of the Redatam package.

work directory  This is the default directory for any files that you produce in winR+ without indicating a specific directory. The Workspace is normally also kept in your work directory. Any name can be used for this directory, but in English it is usually simply called “work”.
A Workspace is the window in winR+ that lists your Command Sets, Selection Sets, AREALISTS and Map Definitions. The Workspace .mdb file that holds this information, also contains your own copy of the data dictionary for the winR+ database that you are using. The Workspace is normally kept in your work directory.

Note that, on your computer, you can create various Workspaces with different names for the same winR+ database, e.g., if more than one person is using the computer or if you have different projects with the same data. Each Workspace has its own copy of the database dictionary.

xBASE
In winR+, used interchangeably with dBASE. See dBASE

.xls
Extension used for a file of a Microsoft Excel spreadsheet.
INDEX

NOTE: winR+ command keywords are always in upper case letters, e.g., TABLE.
Frequently used keywords, introduced in the text of the Manual are indexed here. All
keywords are in the Reference Guide to the winR+ Command Language (Appendix
B) and are also indexed here. However, it is suggested that you use the keyword index
of the Reference Guide, page 224, when looking for detailed information on specific
keywords.

Abort run, 70
AccessPlan, 9, 210
advanced consistency checks, 142
age groups, 110
aggregate variables at entity level, 99
group, 267
alias, 104, 255
ALL, 77, 154, 229
append
source AREALIST and target file, 198
Arc/Info, 197
Arc/Info®, 268
ArcView, 197
ArcView®, 268
AREABREAK, 75
Output report window, 76
Summary table, 76
AREALIST, 26, 95, 162, 267
field, 101
rows, 162
adding a new column, 173
alias, 104
Browser (see: AREALIST browser), 101
Classification of a variable, 188
column statistics, 166
columns, 101, 162
compile error if already exists, 105
complete path, 186
display selected records on map, 192
export, 167
export to spreadsheet, 107
export to target system, 197
field name rules, 172
fields, 101, 162
file types, 106
import, 167
link to GIS, 196
Linking to map definition, 185
linking two AREALISTS, 176
names of areas, 148
OUTPUTFILE, 99, 106
OVERWRITE for .dbf, 107
printing, 101
Query expression builder, 168, 180
records, 101, 162
rows, 101
Saving as .dbf, 106, 148
saving records, 168
see also AREALIST Browser, 162
Selecting and Querying areas, 193
selecting areas, 178
selection of records, 168
selection of records, see AREALIST Browser, 164
sequential numbering of elements, 132
Simple Count, 51
Sorting the list, 102, 164
source AREALIST and target file, 198
source of external variables, 145
to create Selection Set, 177
variables at higher entity level, 106
variables at lower entity level, 103
view, 101, 107
window (Fig.), 105
AREALIST Browser, 101, 162
Desel ecting an area, 102, 164
line cursor, 102, 163
selecting an area, 102, 164
sorting, 102, 164
adding a new column, 173
AREALIST name, 163
Changing column width, 165
changing row height, 163
column statistics, 166
display selected records on map, 192
editing a column, 166
hide/reveal columns, 166
inverting selection of records, 164
names of columns, 165
saving selected records, 171
selection of records, 164
show a hidden column, 166
show only selected records, 170
sorting records, 164
temporarily store a selection, 170
AS
with logical expression, 124
with logical expression, caution, 125
ASCII, 268
Assist, 27, 62
accessing in Windows 3.x and 95, 64
AREABREAK, 75
AVERAGE Command Assist, 69
DEFINE Command Assist, 80
inging changes, 65
expression builder (Fig.), 79
for new and occasional users, 62
gen eral Command Assist, 64
gen eral Command Assist (Fig.), 78
guided Assist, 64
R+ Data Link window (Fig.), 199
RUNDEF Command Assist, 64
TABLE Command Assist, 65, 68
Assist windows, 267
asterisk *
 comment in command language, 76
AVERAGE, 26, 69
tographing, 47
Output report window, 54
backup, 36
bin variable, 153
bna, 268, 270
bna file, 184
Boolean variables, 126
branches
database, 23
categories, 24, 268
CELADE, xxi, 9, 210, 268
 Internet address, xxi
CEPAL, 269
CIDA, 268
classes, 189, 268
Classification and Mapping window, 188
Classification of a variable, 188
calculate values of new column, 174
color ramp , 189
colors, 189
equal intervals, 189
manual, 175, 189
number of classes, 176
quantiles, 189
quantiles, 175
single value, 188
classification type
manual, 189
Classification and Mapping
window (Fig.), 191
click, 268
CmdSet, 25
cf. command set, 60
CODEBOOK, 89, 244
color ramp, 189
columns, 268, 271
see variables, fields, 165
combo box, 269
Command Editor window, 62
widening the window, 68
command language
de-activate commands via *, 76
improving readability, 75
comment with asterisk *, 218
comment with aterisk *, 75
continuation character (none), 68
De-activate keyword with asterisk *, 75
DEFINE, 79
layout for readability, 68
Reference Guide, 217
RUNDEF, 63
see: winR+ language, 59
structure, 60
TABLE, 65
command set, 5, 25
external database variables, 143
import, 142
keywords in, 60
to generate Selection Set, 136
command set algorithms
age groups by formula, 110
age of oldest person in household, 119
age of youngest person in household, 123
average age of persons in household, 117
children ever born, 126
household overcrowding, 124
identifying sequences of houses, 133
number of generations in household, 113
number of items in household, 130
old persons living alone, 157
sequential numbering of elements, 132, 155
sex of oldest person, 121
Command Sets in winR+ and R+, 211
comment, 76, 219, 226
compile, 65, 69, 78
testing each command, 79
Compile button, 69
compiling, 25
compression, 1
continuation character (none), 68
conversion
R+ to winR+ database, 138
correction program, 59
COUNT
COUNT vs. TOTCASES, 159
Counting elements
Simple Count, 51
CROSSTABS, 26, 68
entity levels different, 92
Output with 3 or 4 variables, 87
percents, 49
row and column variables, 49
CrossTabulation
EasyTabs, 48
EasyTabs window (Fig.), 48
Customization, 16
options window (Fig.), 17
output page, 70
CUT and PASTE, 86
Data Link, 7, 196
data protection, 8
Data security, 8, 23
database
branches, 23
quality, 116, 126
database concepts, 21
database structure, 24
  database hierarchy (Fig.), 22
  definition, 24
  window, 36
dBAGE variables
  defined in the dictionary, 148
dBASE®, 269
DBF, 255, 269
  AREALIST, 106
  AREALIST viewing, 107
  export, 167
  impport, 167
DBF variables
  defined in the dictionary, 148
DECIMAL, 155
DECIMALS, 242
decision support system tools, 9
DEFAULT, 98, 242, 269
default map color, 184
DEFAULT values, 115
DEFINE, 79, 80, 230
  introduction to, 59, 218
  OPTIONS DEFAULT, 91
  RANGE, 85
  SAVE, 153
  TYPE, 85
  VALUE LABELS, 84
  VARG LABELE, 84
DEFINE...AS, 231
DEFINE...AS COUNT, 232
DEFINE...AS DATASET, 144, 233
DEFINE...AS RECODE, 82, 236
DEFINE...AS SUM, 238
DEFINE...AS TOTCASES, 239
DEFINE...FOR, 83, 240
DEFINE...LIKE, 241
DEFINE...OPTIONS, 242
DEFINE...OPTIONS DECIMALS, 244
DEFINE...OPTIONS DEFAULT, 244
DEFINE...OPTIONS MISSING, 244
DEFINE...OPTIONS NOTAPPLICABLE, 244
DEFINE...RANGE, 244
DEFINE...RANGE CODEBOOK, 245
DEFINE...SAVE, 245
DEFINE...TYPE, 247
DEFINE...VALUE LABELS, 248
DEFINE...VARIABLE LABEL, 249
Delete
  SAVEEd variable, 156
demonstration database, 10
DFT files, 142
dictionary, 24, 44
  definition, 24
  deleting new variables, 152
  editing external variables, 152
  expand/collapse branches, 37
  variable window, 36
  window (Fig.), 38
directory, 269
  work, 138
Display the map, 184
documents, 34
DOS, 10, 269
EasyTabs, 44
  Average, 51
  CrossTabulation, 48
  Frequency, 45
  Frequency window (Fig.), 45
  graph output, 47
  limitations, 45, 50
  only dictionary variables, 44, 45
  Save, 46
  Simple Count, 51
ECLAC, 269
EduPlan, 9, 210
element, 269
ELSE, 83
Entities
  identifiable, 8
  entity, 269
  branches, 22
  definition, 22
  identifiable, 8
  root, 22
  selectable, 8
  unselectable, 8
entity elements, 22
  definition, 22
INDEX — 277

select/deselect, 41
selectable, 23
unselectable, 23
Entity. Variable, 269
Entity.Name. VariableName, 24, 58, 219, 269
equal intervals, 189
equal intervals, 175
Excel, 94, 107
Excel®, 269
exiting winR+, 16
Export
AREALIST to .dbf, 167
AREALIST to spreadsheet, 107
AREALIST to target system, 197
command sets, 142
selection set, 142
expression
evaluated as 0 or 1, 130
logical with AS, 124
True or False in CmdSet, 124
expression builder, 53, 79
calculate values of new column, 174
for AREALIST record selection, 168
temporarily store a selection, 170
window (Fig.), 80
external database variables, 143, 270
defined in the dictionary, 148
from an AREALIST, 145
permit easy updating, 149
external variables, 270
FIELD, 147
field name rules, 172
fields, 101, 162, 270, 271
external database, 149
see also variables, 162
Figures in this Manual, xv
file structure, 153
file structure of winR+, 153
files
transposed, 153
filtering cases
FOR, 74, 83
RUNDEF...UNIVERSE, 72
Folder, 270
font
map display, 190
font:, 71
Footer of a page, 70
FREQUENCY, 26, 66
EasyTabs, 45
graphing, 47
of REAL variable, 247
REAL variable, 98
General Command Assist
see: Assist, 62
GENERATE in R+, 154
Geographic Information System, 270
Geographical Information Systems (GIS), 2, 7
geographical structure, 1
GIS, 270
data Link, 196
GIS file created in R+, 198
Glossary, 267
graph output, 47
crosstabulations, 50
group, 77
Guided Assists
see: Assist, 62
Header of a page, 70
Help, xiv, 37
hierarchical geographical structure, 1
hierarchical processing, 51
advanced, 156
DEFINE...AS COUNT, 88
description, 6
how it works in winR+, 91
limitations, 116, 120, 156
one pass only, 156
Quasi pre-processing, 156
TOTCASES, 157
virtual multiple-pass processing, 161
hypertext links, 70
icon bar (Fig.), 32
icons (Figs.), 32
IDRC, 9, 270
Import
.dbf to AREALIST, 167
AREALIST from other Workspace, 167
command sets, 142
selection set, 142
workspace, database, documents, 140
inconsistencies
between the source and target, 201
detecting between source and target files, 198
Redflag Control Field, 200
inheritance of properties, 115
initialization for new database, Workspace, 142
Installing winR+, 15
Readme file, 34
Windows 3.x and 95, 15
integer division, 111, 118
Internet
CELADE page, xxii
job, 271
key field, 178
keyboard
CTRL key for multiple selection, 35
KEYFIELD, 144, 177
keywords, 270
keywords (commands)
Reference Guide, 217
[lookup individual keywords], 59, 218
label, 44
Language change, 18
launching winR+, 16
legend, 190
legend (map), 190
LIKE, 115, 230, 242
Link Forms, 186
Link to Map, 188
Linking source and target files, 196
linking two AREALISTS, 176
logo, winR+, 16
Lotus 1-2-3®, 270
main menu, 270
main menu (bar), 31
main screen (Fig.), 31
manual
conventions, xv
how to use, xii
notation, xvi
symbols, xv
Typographic conventions, xvi
Typographical conventions, xvi
map
default colors, 184
fonts, 190
title, comments, etc., 190
colors, 189
default color set, 184
legend, 190
map definition, 183, 270
Linking to AREALIST, 185
Map Definition Editor
window (Fig.), 184
map, display, 184
classify a variable, 187
layout, 193
legend, 190
selected AREALIST records, 192
Selecting and Querying areas, 193
MapInfo, 209
mdb, 270
mdb file, 27
metadata, 24
microdata, 2
Microsoft Access®, 270
.mif/.mid, 183
minimum and maximum values
rule for classes, 189
MISSING, 230, 242, 262
Missing values
in the Range (warning), 37
missing values, include, 46
Multilingual facilities, 8
multiple selection, 35, 143
nacimien.dbf, 145
New Miranda, xvi, 19, 23, 30, 117
English, Spanish Workspaces, 34
multi-branch database (Fig.), 25
NOKEY, 262, 263
NOT APPLICABLE values, 115
NOTAPPLICABLE, 230, 242
NotApplicable values
in the Range (warning), 37
notation, 223
notation used in the manual, xv
operators, 112
OPTIONS
MISSING, 262
NOKEY, 262
OVERWRITE, 262
PCC, 262
PCR, 262
PCT, 262
TITLE, 262
WEIGHT, 262
ZERO, 262
OPTIONS (DEFINE)
DECIMALS, 242
DEFAULT, 242
MISSING, 242
NOTAPPLICABLE, 242
OPTIONS DECIMAL, 155
OPTIONS DEFAULT, 91
Output report
AREABREAK, 76
AREABREAK hypertext (Fig.), 76
AREALIST, view, 101
for command editor (Fig.), 72
hypertext links, 54
list of Tables, 54
window (Fig.), 55
OUTPUTFILE
AREALIST, 99
Outputs, 7
OVERIDE, 230
OVERWRITE, 107, 254, 255, 262-265
page layout, 47, 70
page orientation, 71
page setup, 47, 70
password for Work mode, 19
PAT file, 197
PCC, 262
PCH, 263
PCR, 262
PCT, 262, 263
PCV, 263
percentages
Easy CrossTabs, 49

PCR, PCC, PCT, 262
Point and Shoot
to identify areas on a map, 194
preferences, 16
primary key, 176
print preview, 47
print setup, 71
printer, 71
Printing
AREALIST, 101
EasyTabs, 47
Fit to page, 71
graph, monochromatic, 48
Header and Footer, 70
map, 191
Options window (Fig.), 18
output font, 71
output page, 70
page layout, 47
page setup, 47
page setup (Fig.), 71
print preview, 47
protection, 8
quantile, 189
quantiles, 175, 189
Quattro Pro®, 270
Query Expression Builder, 168, 180
Query mode, 194
R+, 271
cf. Redatam-Plus, xv
R+ Data Link
moving AREALIST to external target, 198
window (Fig.), 199
R+GIS Link
see Data Link, 196
RANGE, 85
CODEBOOK, 89
estimated, with CODEBOOK, 90
importance for a crosstabs, 97
incorrect in database (warning), 37
with FOR, 90
Readme file, xiv, 34
RECODE, 82
ELSE, 83
HIGHEST, 83
LOWEST, 83

records
rows of an AREALIST, 101, 162
see rows, 163

Redatam, 1
Redatam, 271
Redatam-Plus, 1
. def file, 139
compared with winR+, xv
conversion of database to winR+, 138

Redatam-Plus©, 271
Redflag Control Field, 200
codes for inconsistency detection, 201

Reference Guide to the Command Keywords, 62, 217
Index to the Command Keywords, 224
looking up keywords, 221
Using the Reference Guide, 222

report of process, 69
Requirements for using winR+, 13
hardware, 13
minimum configuration, 14
software, 13
user knowledge, 13

rows, 271
run, 271
run (also called a "job")
see: Selection Set, 26

Run button, 70
RUNDEF, 63, 227
introduction to, 59, 218
RUNDEF Command Assist, 64
RUNDEF Command Assist (Fig.), 66
UNIVERSE, 72
RUNDEF...FOR, 227
RUNDEF...RUNTITLE, 228
RUNDEF...SELECTION, 228
RUNDEF...UNIVERSE, 72
SAVE, 153, 154
Command Set, 47
Delete SAVED variable, 156
GENERATE in R+, 154
variables to a winR+ database, 153

virtual multiple-pass processing, 161
Save As, 63
saving
TABLE as spreadsheet, 94
scale (printing), 70
SEL files, 142
SELECTION, 256
multiple, 35, 143
selection area, 3
Selection Editor window, 41
Partial selection of an area, 41
Total selection of an area, 41
window (Fig.), 40

selection set, 3, 271
created by a command set, 181
ALL, 46, 77, 154
created from map, 194
created from map (Fig.), 196
creation in multi-branch database, 42
export, 142
external file, 182
don from an AREALIST, 136
don from command set, 136
generated from an AREALIST, 177
group, 77
import, 142
non-geographical, 4
planning area (Fig.), 3
selection file (DOS), 182

SELFILE, 182, 229
SELSET, 229, 271
cf. Selection Set, 25
SHARE (Windows 3.x), 15
short varlabel, 269, 271
Simple Count, 51

Sorting
AREALIST, 102, 164
spc, 271
Speed of processing, 7
spreadsheet
from AREALIST, 107
saving a table as, 94
SQL expression builder, 180
statistical process report, 69
INDEX

COUNT vs. TOTCASES, 159
Use with caution, 157
TourPlan, 9, 210
translation, 8
True or False expression in CmdSet, 124
Tutorials
No. 1, 30
No. 2, 39
No. 3, 58
No. 4, 95
TYPE, 85
INTEGER, 86, 111
REAL, 86
REAL, and DEFAULT, 98
REAL, division by zero, 98
Typographic conventions, xvi, 223
UNIVERSE, 72
University of Waterloo, 9, 210, 271
users of winR+, 10
Assist for new and occasional users, 6, 62
UWaterloo, 271
VALUELABELS, 84
variable, 271
variable names
upper and lower case letters, 98
variables, 219
aggregated, 96
alias, 104
combining, 128
definition, 24
dependency, 58
external, 143
identification of, 58
see also fields, 162
short varlabel, 59
statistic, 166, 188
Temporary in a DEFINE, 86
unique-sums algorithm, 130
upper and lower case letters, 98
Yes/No, 130
VARLABEL, 84, 269, 271
Version number of winR+, 38
view a table, 47, 70
hypertext links, 70

statistics
of AREALIST column, 166
statistics of AREALIST variables, 188
status bar, 31
SUM, 117
syntax check
sec: compile, 65
System requirements
System requirements, 13
TABLE, 65, 250
CROSSTABS, 68
AREALIST, 95
AVERAGE, 69, 86
FOR, 74
FREQUENCY, 66
introduction to, 59, 218
spreadsheet, save as, 94
TABLE Command Assist, 65
TABLE...AREABREAK, 75, 251
TABLE...AS, 252
TABLE...AS AREALIST OF, 252
TABLE...AS AVERAGE OF, 256
TABLE...AS CROSSTABS OF, 258
TABLE...AS FREQUENCY OF, 259
TABLE...FOR, 74, 261
TABLE...OF, 261
TABLE...OPTIONS, 261
MISSING, 262
NOKEY, 262
OVERWRITE, 262
PCC, 262
PCR, 262
PCT, 262
TITLE, 262
WEIGHT, 262
ZERO, 262
tables, 26
target population
creating a SelSet, 171
temporary storage of selected records, 170
TITLE, 263
title bar, 271
tool bar, 32
TOTCASES
view a Workspace document, 35
WEIGHT, 126, 263
Windows 3.1, 13
Windows 95, 13
winR+  
   About, 38
   Customization, 16
   data security, 8
   description, xi
   Help, 37
   Installing, 15
   Launching, 16
   logo, 16
   main screen, 30, 31
   main screen (Fig.), 31
   multilingual facilities, 8
   new for R+ users, 203
   obtaining further information, xxi
   open system, 6
   requirements for using, 13
   Speed of processing, 7
   version and revision numbers, xiv, 38
winR+ building blocks  
   description, 9
winR+ Data Link, 196, 198
   description, 7
winR+ database  
   bin variables, 153
   concepts, 21
   creation of, 10
   Delete SAVEd variable, 156
   import, 140
   master, 156
   non-geographical criteria, 4
winR+ file structure, 153
winR+ language, 219
   Basic commands, 59, 218
   Command structure and syntax summary, 219
   conversion program from R+ not available, 59
   description, 5
   introduction to, 59
   Reference Guide, 217
   see Command language, 217
winR+ latest, 37
winR+ map display (optional module), 183
winR+GIS decision support system tools, 9
winR+©, 271
wizard, see: Assist, 27
work directory, 138, 271
Work mode, 19
workspace, 26, 33, 255, 272
   Advanced|Consistency checks, 142
   advantages of, 2
   AREALIST, 99
   AREALIST, view, 101
   BACKUP, 36
   column width adjustment, 35
   creating new, 140
   creation of (Fig.), 138
   definition, 26
   document, 2, 34
   import, 140
   import and export, 142
   map definition, 183
   Microsoft Access database, 138
   New Miranda: English, Spanish, 34
   row width adjustment, 35
   window, 33
   window (Fig.), 33
   window, minimize/maximize, 35
   window, move, 34
WRITE command in R+, 103
xls, 272
xls format, 94
ZERO, 263
zeros, include, 46
ZonPlan, 9, 210