STATA Programs for Using the Intermediate Data Structure (IDS) to Construct Files for Statistical Analysis

Quaranta, Luciana

Published in:
Historical Life Course Studies

2016

HISTORICAL LIFE COURSE STUDIES

VOLUME 3

2016

EHPS NETWORK

MISSION STATEMENT

HISTORICAL LIFE COURSE STUDIES

Historical Life Course Studies is the electronic journal of the *European Historical Population Samples Network* (EHPS-Net). The journal is the primary publishing outlet for research involved in the conversion of existing European and non-European large historical demographic databases into a common format, the Intermediate Data Structure, and for studies based on these databases. The journal publishes both methodological and substantive research articles.

Methodological Articles
This section includes methodological articles that describe all forms of data handling involving large historical databases, including extensive descriptions of new or existing databases, syntax, algorithms and extraction programs. Authors are encouraged to share their syntaxes, applications and other forms of software presented in their article, if pertinent, on the EHPS-Net website.

Research articles
This section includes substantive articles reporting the results of comparative longitudinal studies that are demographic and historical in nature, and that are based on micro-data from large historical databases.

Historical Life Course Studies is a no-fee double-blind, peer-reviewed open-access journal supported by the European Science Foundation (ESF, http://www.esf.org), the Scientific Research Network of Historical Demography (FWO Flanders, http://www.historicaldemography.be) and the International Institute of Social History Amsterdam (IISH, http://socialhistory.org/). Manuscripts are reviewed by the editors, members of the editorial and scientific boards, and by external reviewers. All journal content is freely available on the internet at http://www.ehps-net.eu/journal.

Editors: Koen Matthijs & Paul Puschmann
Family and Population Studies
KU Leuven, Belgium
hislives@kuleuven.be

The European Science Foundation (ESF) provides a platform for its Member Organisations to advance science and explore new directions for research at the European level. Established in 1974 as an independent non-governmental organisation, the ESF currently serves 78 Member Organisations across 30 countries. EHPS-Net is an ESF Research Networking Programme.

The European Historical Population Samples Network (EHPS-net) brings together scholars to create a common format for databases containing non-aggregated information on persons, families and households. The aim is to form an integrated and joint interface between many European and non-European databases to stimulate comparative research on the micro-level.
Visit: http://www.ehps-net.eu.

HISTORICAL LIFE COURSE STUDIES
VOLUME 3 (2016), 01-19 published 29-02-2016

STATA Programs for Using the Intermediate Data Structure (IDS) to Construct Files for Statistical Analysis

Luciana Quaranta
Lund University

ABSTRACT

The Intermediate Data Structure (IDS) provides a common structure for storing and sharing historical demographic data. The structure also facilitates the construction of different open-access software to extract information from these tables and construct new variables. The article *Using the Intermediate Data Structure (IDS) to Construct Files for Analysis* (Quaranta 2015) presented a series of concepts and programs that allow the user to construct a rectangular episodes file for longitudinal statistical analysis using data stored in the IDS. The current article discusses, in detail, each of these programs, describing their technicalities, structure and syntax, and also explaining how they can be used.

Keywords: Intermediate Data Structure, IDS, Historical Demography, Demography, Life Courses, Social History, History, Episodes Table, Survival Analysis, Event History Analysis, STATA

e-ISSN: 2352-6343
PID article: http://hdl.handle.net/10622/23526343-2016-0001?locatt=view:master
PID Extended IDS Software: http://hdl.handle.net/10622/23526343-2016-0002?locatt=view:master

The article can be downloaded from here.

1 INTRODUCTION

The Intermediate Data Structure (IDS) was developed as a strategy aimed at simplifying the collecting and sharing of longitudinal demographic data (Alter & Mandemakers 2014; Alter, Mandemakers & Gutmann 2009). The structure provides a common platform to store data from different databases, regardless of their original form of storage. Further development to such structure is being made by members of the European Historical Population Samples Network (EHPS-Net), who are also creating open access programs to use the IDS to construct files for analysis. This article is written within such initiative. The format of the tables included in the IDS and the way in which data should be stored in them has been discussed in detail in previous works (Alter & Mandemakers 2014), but not much information has yet been provided about how to extract data out of the IDS to produce files for research.

Data stored in the IDS follows the Entity Attribute Data Model (Stead, Hammond & Straube 1982), and therefore contains one attribute per each record, meaning that each row of the table includes one declaration of a Value of a variable Type. Historical demographic data obtained from population registers or family reconstitutions is generally analyzed statistically by using event history analysis, for example, Cox models (Cox 1972; Therneau & Grambsch 2000). To conduct these types of analyses, the data must be formatted as a rectangular episodes table. An episodes table can contain many rows – or spells – per individual. Spells are periods of time when time-varying variables remain constant and, at the end of which, events can take place. Each row in the table of spells begins with a date of change in any variable or event and ends with the next date of change in any variable or event. Given the complexity of longitudinal data, creating episodes files requires advanced data management skills. Although there are various statistical software available to conduct advanced statistical modelling, no programs exist which can be used to automatically setup longitudinal micro-level demographic data in the correct format for analysis. The use of this type of data is therefore restricted to researchers with extensive programming skills, limiting the scope of historical demographic research.

The article *Using the Intermediate Data Structure (IDS) to Construct Files for Statistical Analysis* (Quaranta 2015) presented a series of concepts that allow researchers to use the data stored in the IDS to construct a rectangular episodes file for longitudinal statistical analysis. While presenting these concepts, an extension to the IDS, the Extended Intermediate Data Structure (EIDS), was also introduced. Such extension includes the tables INDIVIDUAL_EXT and CONTEXT_EXT, which have the same structure as the IDS tables, but store variables constructed from calculations or elaborations of source data stored in the IDS. Such variables were defined as extended variables. The article also proposed a format for data extractions and for the output produced by open-access extraction programs. It consists of a *Chronicle file* containing one row per each declaration of any variable or event to be included in the analysis and a *Variable setup file* describing the information contained in such extraction. The article also presented seven STATA programs which can be used directly by researchers to create a file for analysis from the IDS without the need to conduct any further data management or programming.

The current work presents these programs more in-depth. After providing an overview of them in section 2, each program is discussed in detail, describing its technicalities, structure and syntax. In the final chapter an example is given in order to explain how the programs can be used and the output produced by them.

2 OVERVIEW OF THE PROGRAMS

The program **extended IDS table maker** constructs empty tables for the EIDS. Externally created data can be inserted into the INDIVIDUAL_EXT, CONTEXT_EXT and METADATA tables using the program **import data**. **Household size** is an example of a program aimed at creating extended variables at the context level. The data for analysis can be selected from the IDS and the EIDS tables using the program **select type**. These variables can be added to an extraction file using the programs **append individual variables** and **append contextual variables**. Finally, using the program **episodes file creator (EFC)**, data extractions can be converted into a rectangular episodes table that can be used directly for statistical analysis.

Table 1 contains a short description of these programs, their dependency with other programs and the tables which are used. By running the programs in the order in which they are presented in this article,

a rectangular episodes file for analysis can be obtained without conducting any additional data management. Several of the programs can also be used independently of the rest and/or in combination with other open-access or locally developed programs.

Table 1 *Description of seven programs written for STATA to construct files for analysis from data stored in the IDS*

Program name	Description	Dependency with other programs	Tables used by the program
Extended IDS table maker	Creates empty tables for EIDS (INDIVIDUAL_EXT, CONTEXT_EXT, EventChronicle, VarSetup).	None	None
Import data	Imports externally created extended variables and appends them to INDIVIDUAL_EXT and CONTEXT_EXT.	Extended IDS table maker	INDIVIDUAL_EXT, CONTEXT_EXT, METADATA; external imported data
Household size	Example of a program that creates a contextual level extended variable.	Extended IDS table maker	INDIV_CONTEXT, CONTEXT, CONTEXT_EXT, METADATA
Select type	Creates Excel tables containing a list of all unique variables types stored in the INDIVIDUAL, INDIVIDUAL_EXT, CONTEXT and CONTEXT_EXT tables to allow the user to select variables for analysis.	Extended IDS table maker	INDIVIDUAL, INDIVIDUAL_EXT, CONTEXT, CONTEXT_EXT
Append individual variables	Appends selected individual variables to the Chronicle and Variable Setup files.	Extended IDS table maker, Select type	INDIVIDUAL, INDIVIDUAL_EXT, INDIVIDUAL_SELECT, INDIVIDUAL_EXT_SELECT, Chronicle, VarSetup
Append contextual variables	Appends selected contextual variables to the Event Chronicle and Variable Setup files.	Extended IDS table maker, Select type	CONTEXT; CONTEXT_EXT, CONTEXT_SELECT, CONTEXT_EXT_SELECT, Chronicle, VarSetup
Episodes file creator	Rectangularizes and reformats the information contained in the data extraction, giving an episodes table during output, which is a rectangular file that is ready for statistical analysis.	Can be used independently or in combination with all of the other programs	Chronicle, VarSetup

The seven programs presented are saved as *.ado* files and are accompanied by a user's guide (a help file) saved as a *.sthlp* file. Programs and help files can be run from the command line in STATA in the same way as any other built-in STATA program. The programs and help files are included as attachments to this article[1]. The attachment also includes an Excel file containing a demo database stored in the IDS, which can be used to test these programs. The information contained in the tables of this database was made up, but it was created to resemble data obtained from population registers. The files *UseIDSprograms_example.do* and *RunPrograms.log* are also attached to this article. The STATA do file is a command file containing an example of how the seven programs can be used to create an episodes table from the demo database. The log file shows the same example also including the output produced in the result window of STATA when running each program. This example is discussed in more detail in section 10 of this article, in order to provide a step-by-step guide of how to use the programs and of their output.

Prior to running the programs, the working directory needs to be set in STATA, and the tables INDIVIDUAL, INDIV_INDIV, CONTEXT, CONTEXT_CONTEXT, INDIV_CONTEXT and METADATA must be saved as *.dta* files in such folder. Tables stored as text or Excel files can be imported into STATA and saved as data files to be used by the programs with the command "**import**". Alternatively, ODBC sources can be directly loaded into STATA by using the command "**odbc**". The personal system directory also needs to be set, saving all programs and help files in this directory[2]. Although the programs are specifically written to be used in STATA, it is possible to export the data outputs to conduct statistical analysis with other software. Members of the EHPS-Net are also working on making a version of the program **EFC** for R

1 The programs and future updates of them will also be published in the Repository of the EHPS-Net Webpage.

2 The working and personal system directory can be set by, for example: **cd "C:\IDS\DTA files"** and **sysdir set PERSONAL "C:\IDS\IDS software"**. Users working in a restricted environment where the personal system directory cannot be set can instead use, for example, the following syntax: **ado + "C:\IDS\IDS software".**

The **EFC** is the largest and most important program introduced in this work. It can be used after running the other six programs, independently of them and/or in combination with other local or open-access programs. The **EFC** converts data extractions into rectangular episodes tables that are ready for statistical analysis.

As was explained in Quaranta (2015), to use the **EFC** data, extractions have to be saved in a *Chronicle file* (saved using the name *Chronicle.dta*) and they have to be accompanied by a *Variable setup file* containing information on each extracted variable (saved using the name *VarSetup.dta*). The *Chronicle file* must contain the fields Id_I, Type, Value, Day, Month, Year, and DayFrac, as is shown in Table 2. DayFrac corresponds to a fraction of a day, and it is used by the program to adjust date collisions. Date collisions occur when there is more than one change in the Value of a Type on the same date. By adding a fraction of a day, these Values can be included in the episodes table in the correct chronological order. The *Variable setup file* must contain the fields Type, Transition and Duration, as is shown in Table 3. Transition distinguishes whether a Type is a time-varying variable with Values that change at the beginning of a spell (Transition = Start), an event occurring at the end of a spell (Transition = End) or a time-invariant variable (Transition = Invariant). This information allows linking variable Values correctly to dates when building the episodes table. Duration is used to specify whether the Values of a variable are only valid on their date of declaration (Duration = Instant) or whether they are valid during the period elapsing between two consecutive dates of declaration or between the last date of declaration of the Type and the last date of observation for the individual (Duration = Continuous).

Table 2 *Description of the fields contained in the Chronicle file*

Name of column	Description
Id_I	Identifying number of each individual in the data
Type	Variable name
Value	Variable value
Day	Day when the variable changes value or the events takes place
Month	Month when the variable changes value or the events takes place
Year	Year when the variable changes value or the events takes place
DayFrac	Fraction of a day, which must be assigned in cases when there is more than one Value of a specific Type on the same date. This field is used to sort Values correctly in a chronological order, and therefore DayFrac should be greater for changes in Values that take place later.

Table 3 *Description of the fields contained in the Variable setup file*

Name of column	Description
Type	Variable name
Transition	Distinguishes whether the Type is a time-varying variable that changes value at the start of the spell (Transition = Start), an event that changes value at the end of the spell (Transition = End) or a time-invariant variable (Transition = Invariant).
Duration	Distinguishes whether the Values of a Type are valid only on their date of declaration (Duration = Instant) or between a date of declaration and the next date of declaration/End_date (Duration = Continuous).

3 PROGRAM EXTENDED IDS TABLE MAKER

The aim of the program **Extended IDS table maker** is to create empty EIDS tables. By using this program, it is possible to create the EIDS tables INDIVIDUAL_EXT and CONTEXT_EXT, and the tables Chronicle and VarSetup, which are used to construct episodes files. The columns contained in each of the EIDS tables are the same as those contained in the IDS tables. The only difference is the time stamp. Besides Day, Month and Year (and Start and End Day, Month and Year), the timestamp also includes the field DayFrac.

The program can be used to create one or several tables at the same time, by using the syntax shown

below. The newly created tables replace any already existing files previously saved with the same name. The program is saved as the file *IDSextended.ado* and it is accompanied by the help file *IDSextended.sthlp*.

Syntax
IDSextended, [, options]

Options:

individual(1): Creates the table INDIVIDUAL_EXT.dta.

context(1): Creates the table CONTEXT_EXT.dta.

Chronicle(1): Creates the table Chronicle.dta.

varSetup(1): Creates the table VarSetup.dta.

4 PROGRAM IMPORT DATA

Import data is a program that allows users to import extended variables created by the researcher or by other open-access software and to append them to the INDIVIDUAL_EXT and CONTEXT_EXT tables. It also appends metadata relating to such variables to the METADATA table. The program is saved as the file *importData.ado* and it is also accompanied by the help file, *importData.sthlp*.

In the syntax of the program it is necessary to specify the name of the external file containing the data to be imported (source data), and the name of the external table containing metadata specific to the imported variables (source metadata). It is also necessary to specify the table to which the external data should be appended (destination), which can be the INDIVIDUAL_EXT or CONTEXT_EXT tables. The source data, source metadata, destination and METADATA tables must already exist in the working directory, and be saved as STATA data files (*.dta*). All imported variable Types included in the source file must also be included in the source metadata file in order for later programs to work.

The first part of the program controls whether the source and destination files exist. The second part of the program checks whether the source data and metadata tables contain the correct fields. If problems are encountered in either of these steps, an error message is given and the program is stopped. The third part of the program drops any fields of the source data and metadata tables that are not relevant to the destination table. The fourth part of the program appends the source data to the destination table and the source metadata to the METADATA table.

Syntax
importData *sourceData sourceMetaData destination*

Specifications:
sourceData: name of external .dta source file containing the data to be imported.

sourceMetaData: name of external .dta source file containing the metadata specific to the variable Types imported.

destination: name of the destination file. It can either be the INDIVIDUAL_EXT or CONTEXT_EXT table.

5 PROGRAM HOUSEHOLD SIZE

Household size is an example of a program aimed at creating a contextual-level extended variable by using the IDS tables during input. The program is saved as the file *householdSize.ado* and it is also accompanied by the help file *householdSize.sthlp*. The size of the household is calculated by taking into consideration the Context identifiers. The Values are stored by the program in the CONTEXT_EXT table. This variable can be linked to each individual living in the household by using the program **append contextual variables.**

This program uses the files *INDIV_CONTEXT.dta*, *CONTEXT.dta*, *CONTEXT_EXT.dta* and *METADATA.dta*. Part 1 of the program checks whether such files are available in the working directory, and if any of them is missing an error message is given and the program is stopped.

Part 2 of the program reads during input the CONTEXT table and selects all Context identifiers where the Type is equal to "Level" and the Value is equal to "Household". The selected Context identifiers are merged to the INDIV_CONTEXT table in order to select all dates of entry and exit of individuals in each household.

Part 3 of the program uses the above selected information to calculate household size. All dates when there was any entry or exit of at least one individual to or from the household are first identified. In a series of steps the total household size at each of these dates is calculated.

Part 4 of the program appends the created information to the CONTEXT_EXT table. Id_C is set to the corresponding household ID. The Type is set to "Household_size" and Source to "household_size_program_v1". All unique dates and the calculated household sizes are appended to the date and Value fields respectively.

Part 5 of the program appends information relating to the newly created variable to the METADATA table. The Type field is set to "Household_size", the Value field is set to "Definition", the Extract field is set to "household_size_program_v1" and the Type_T field is set to "CONTEXT_EXT".

As is shown below, the syntax of this program is very simple, and just contains the program name. No additional specifications are needed or options need to be set.

Syntax
householdSize

6 PROGRAM SELECT TYPE

The program **select type** has the aim of producing Excel tables containing a list of all unique variable Types that are stored in the INDIVIDUAL, INDIVIDUAL_EXT, CONTEXT or CONTEXT_EXT tables, in order to allow the researcher to select the variables required for analysis from the full set of variables available in these tables. The program can be used to create an Excel file for one or more of the IDS and EIDS tables, by using the syntax specified below. The program is saved as the file *selectType.ado* and it is also accompanied by the help file *selectType.sthlp.*

The program is structured into four sections, each of which relates to the tables INDIVIDUAL, INDIVIDUAL_EXT, CONTEXT or CONTEXT_EXT. Which part of the program is run depends on the options set by the user through the syntax specified below. At least one option must be set, but it is also possible to set all options at the same time in order to create Excel files to select variables from all four tables.

An Excel file containing a list of all unique variable Types stored in the INDIVIDUAL, INDIVIDUAL_EXT, CONTEXT or CONTEXT_EXT tables is created by the program in order to allow researchers to select the desired information. The fields Explanation, Select and Duration are also added to each Excel file. The field Explanation is copied by the program from the METADATA table. The fields Select and Duration are left blank. The researcher must open each Excel file and specify the value 1 in the Select field of each variable Type that they wish to include in the extraction. The field Duration should be set to

Continuous if the values of this variable are valid from the date of declaration until the next date of declaration or until the last date of observation for the individual and to Instant if values are only valid on their date of declaration. The field Duration is used by the program **EFC**. When the value Continuous is assigned to such field, the program fills down missing cells by copying the Value of the same Type and Id_I from the row that precedes in chronological order.

After entering the desired information in the fields Select and Duration, the researcher should save and close the Excel files. The programs **append contextual variables** and **append individual variables** use these Excel files during input.

Syntax
selectType, [options]

Options:

individual(1): Creates the file INDIVIDUAL_SELECT.xls, which contains a list of unique Types stored in the INDIVIDUAL table.

individualExt(1): Creates the file INDIVIDUAL_EXT_SELECT.xls, which contains a list of unique Types stored in the INDIVIDUAL_EXT table.

context(1): Creates the file CONTEXT_SELECT.xls, which contains a list of unique Types stored in the CONTEXT table.

contextExt(1): Creates the file CONTEXT_EXT_SELECT.xls, which contains a list of unique Types stored in the CONTEXT_EXT table.

7 PROGRAM APPEND INDIVIDUAL VARIABLES

The program **select type** created the Excel files *INDIVIDUAL_SELECT.xls* and/or *INDIVIDUAL_EXT_SELECT.xls*, which allowed the researcher to select the individual-level variable Types that should be included in the extraction, among those stored in the IDS and the EIDS tables. The program **append individual variables** has the aim of appending the selected individual-level variables to the *Chronicle* and *Variable setup* files. It can be run to append Types from the INDIVIDUAL table by using the syntax **appendIndivVar, table(INDIVIDUAL)** or from the INDIVIDUAL_EXT table by using the syntax **appendIndivVar, table(INDIVIDUAL_EXT).** The program is saved as the file *appendIndivVar.ado* and it is also accompanied by the help file *appendIndivVar.sthlp*.

Part 1 of the program checks that all the files used are available in the working directory. The program uses the files *INDIVIDUAL.dta* and *INDIVIDUAL_SELECT.xls*, or the files *INDIVIDUAL_EXT.dta* and *INDIVIDUAL_EXT_SELECT.xls*, depending on the syntax specified by the researcher. It also uses the files *VarSetup.dta* and *EventChronicle.dta*. An error message is given and the program is stopped in case any of these files are not found in the working directory.

Part 2 of the program checks whether at least one Type was selected (Select = 1) in the file *INDIVIDUAL_SELECT.xls* (or *INDIVIDUAL_EXT_SELECT.xls*). If no types were selected by the user an error message is given and the program is stopped.

Part 3 of the program appends all changes in the Values of selected individual variable Types to the *Chronicle file*. Changes in the values of selected variable Types are obtained from the INDIVIDUAL table (or INDIVIDUAL_EXT). The program selects changes in Values of Types that are stored either using a single date field (Day, Month, Year) or start and end date fields (Start_day, Start_month, Start_year, End_day, End_month, End_year). Values of Types stored using start and end date fields are assigned to the individual on each start date. The program also assigns a Value -1 on the final end date for which a variable Value is recorded, as well as on end dates after which there is a gap in the data for such Type and Id_I.

The INDIVIDUAL and INDIVIDUAL_EXT tables allow storing variable Values which are of a contextual nature. For these kinds of variables the field Value_Id_C is used and the attribute Value remains empty. For example, for the variable "Birth_location" the Value attribute is left empty and the field Value_Id_C contains the identifier of the context in which the individual was born. When appending information to the *Chronicle file* for each of these variable Types, the program assigns the names of the context to the attribute Value. The program obtains the name of each context by selecting the Type "Name" from the CONTEXT table.

Part 4 of the program appends information on the selected variable Types to the *Variable Setup File*. The fields Type, Duration and Transition are appended. The fields Type and Duration are obtained directly from the Excel files *INDIVIDUAL_SELECT.xls* (or *INDIVIDUAL_EXT_SELECT.xls*), while Transition is created by the program, as described below.

The INDIVIDUAL and INDIVIDUAL_SELECT (or INDIVIDUAL_EXT and INDIVIDUAL_EXT_SELECT) are first merged and only selected variable Types are kept. An error message is given if the field Date_type is not unique across all rows of a specific Type.

The field Transition is used by the program **EFC** to distinguish whether a variable is an event which can occur at the end of a spell (Transition = End), a time-varying variable which changes value at the beginning of the spell (Transition = Start) or a time-invariant variable (Transition = Invariant). The current program assigns the value End to the field Transition for selected Types where the field Date_type was set to Event. To selected Types where the Timestamp was empty in the tables, the program assigns value Invariant to the field Transition. For all other selected Types, Transition is set to Continuous.

Before appending the fields Duration and Transition to the *Variable Setup File*, the program checks that these fields are unique for each variable Type. In case of error a message is given and the program is stopped.

Syntax
appendIndivVar, table(*tabname***)**

Specifications:

*tabname***:** source file (INDIVIDUAL or INDIVIDUAL_EXT) which contains individual variables to append to the Chronicle and Variable setup files.

8 PROGRAM APPEND CONTEXTUAL VARIABLES

The program **Select type** created the Excel files CONTEXT_SELECT.xls and/or CONTEXT_EXT_SELECT.xls, which allowed the researcher to select stored contextual-level variable Types to be included in a data extraction. The program **Append contextual variables** has the aim of appending these selected contextual variables to the *Chronicle* and *Variable setup* files, assigning the values of such variables to individuals during the periods in which they were present in each context. For example, all changes in the value of household size are assigned to all the individuals living in the household at the time of each change. All selected contextual variables are assigned to individuals at the same time by the program. The program can be run to append Types from the CONTEXT table by using the syntax ***appendContextVar, table(CONTEXT)*** or from the CONTEXT_EXT table by using the syntax ***appendContextVar, table(CONTEXT_EXT)***. The program is saved as the file *appendContextVar.ado* and it is also accompanied by the help file *appendContextVar.sthlp*.

Besides the Excel file CONTEXT_SELECT.xls (or CONTEXT_EXT_SELECT.xls, depending on the syntax specified by the researcher), the program also uses the tables INDIV_CONTEXT.dta, *VarSetup.dta* and *EventChronicle.dta*, which must be stored in the working directory. In Part 1 of the program an error message is given and the program is stopped if any of the files used are not found.

In Part 2 of the program the selected Types are appended to the *Variable setup file*. The program assumes that all selected contextual variables are time-varying, and the field Transition is set to Start. This option can be later changed manually in the *Variable setup file*. The field Duration is obtained directly

from the file *CONTEXT_SELECT.xls* or *CONTEXT_EXT_SELECT.xls*.

In Part 3 of the program, Dates of change in the variable Types that were selected are stored in a temporary file. If no Types were selected (i.e. the column Select has no rows that were set to 1) in the files *CONTEXT_SELECT.xls* or *CONTEXT_EXT_SELECT.xls*, an error message is given and the program is stopped. The program selects changes in Values of Types that are stored either using a single date field (Day, Month, Year) or start and end date fields (Start_day, Start_month, Start_year, End_day, End_month, End_year). Values of Types stored using start and end date fields are assigned to the context on each start date. The program also assigns a Value -1 on the final end date for which a variable Value is recorded, as well as on end dates after which there is a gap in the data for such Type and Id_C.

In Part 4 of the program, Values of contextual-level variables are assigned to individuals. The temporary file saved above is used together with the INDIV_CONTEXT table. All changes in the Value of a Type are assigned to individuals on the dates when these changes occurred if the individual was present in the context on the date of change. In other words, these Values are assigned if the date of change of the Value of a Type occurs on or after the Start_date of the individual in the context but before the End_date of the individual in the context. In many cases there is no change in the Value of a Type that occurs on the same date when the individual enters the context (Start_date). In such cases, the Value of the Type which was declared on a date that preceded the Start_date of the individual in the context is assigned to the individual on his or her Start_date but only for types for which the option Duration was set to Continuous. If an individual leaves a context for a period and later returns, the Value of the latest change in a variable Type preceding the date of return into the context is assigned to the individual on such date (also only for cases where Duration was set to continuous).

If there is no declaration of a Value of a Type on a date that coincided or preceded the Start_date of an individual to the context, the Value "NoValue" is temporarily assigned on this Start_date for all variables where the option Duration was set to Continuous. This helps to avoid copying down values of contextual variables for individuals that move from one context to the next when running the program **EFC**. After copying down variable Values, such program replaces NoValue with missing cells and later with -1 for numerical variables.

In Part 5 of the current program the Values of the selected Types that were assigned to individuals in Part 3 are appended to the *Chronicle file*.

Syntax
appendContextVar, table(*tabname***)**

Specifications:

*tabname***:** source file (CONTEXT or CONTEXT_EXT) containing contextual variables to append to the Chronicle and variable setup files.

9 PROGRAM EPISODES FILE CREATOR

The program **EFC** uses during input the *Chronicle file* (saved using the name *Chronicle.dta*), containing the data extraction, and the *Variable setup file* (saved using the name *VarSetup.dta*), containing information relative to such extraction. These files can be created using the program **Extended IDS table maker,** and information can be added to them using the other programs presented in this work. Alternatively, these tables can be created directly by the user, as long as their structure adheres to the IDS standard and to the specifications described in Quaranta (2015) and also briefly outlined in Tables 2 and 3. The program is saved as the file *EpisodesFileCreator.ado*, and it is also accompanied by the help file *EpisodesFileCreator.sthlp*.

To correctly construct episodes tables, each date of change in any of the variables considered in the analysis must be included in the extraction (*Chronicle file)*. The *Chronicle file* should contain the columns Id_I, Type, Value, Day, Month, Year and DayFrac. It must also contain a variable which defines the period in which the individual is at risk of experiencing the event of interest. When running the

program **EFC**, the variable defining the period at risk must be specified in the syntax, as shown below. This variable should have a Value 1 when the individual first becomes at risk and Value 0 when the individual last stops being at risk. Gaps in the period of risk can also be included by specifying the Value 0 on the date when the individual exits from observation and 1 on the date when the individual returns to observation. At the end of the program all spells when the individual is not at risk are deleted from the episodes table.

Information relating to the variables included in the extraction must be included in the *Variable set-up file*. The specifications of such file are described in detail in Quaranta (2015) and are also briefly outlined in Table 3. The file should contain the fields Transition and Duration. As discussed in previous sections of this article, Transition distinguishes between events (which change value at the end of a spell – Transition = End), time-varying variables (which change value at the start of a spell – Transition = Start) and time-invariant variables (Transition = Invariant); Duration specifies whether the values of a Type should be copied down to fill in missing cells.

When creating an episodes table, labels can be added by the program to Values of categorical numerical variables. Such labels can be stored in an external file, which must contain the fields Type, Value and ValueLabel and must be saved as a STATA file in the working directory. The file can be specified in the syntax when running the program **EFC**, as shown below. It is not compulsory to use an external file of labels when running the program.

PART 1 *READING AND PREPARING THE VARIABLE SETUP FILE*

Part 1 of the **EFC** has the aim of reading the *Variable setup file* and preparing smaller temporary files which will be used by the program in order to treat each variable Type correctly when creating the episodes table. If the *Variable setup file* is not found in the working directory, an error message is given and the program is stopped. This part of the program also runs a control which ensures that the *Variable setup file* includes the necessary fields. If the **EFC** is run by specifying a file of labels in the syntax, the program also controls whether such file contains the correct fields. If an error is encountered by any of these controls, an error message is given, specifying which fields are missing, and the program is stopped.

PART 2 *READING AND PREPARING THE CHRONICLE FILE*

Part 2 of the **EFC** has the aim of reading and preparing the *Chronicle file*. If the file is not found or it does not contain the correct fields, error messages are given and the program is stopped. An error message is also given if the at risk variable specified in the program's syntax is not found in the data extraction.

The INDIVIDUAL and INDIVIDUAL_EXT table allow storing variables which are of a date nature, for example "Date_of_birth". The attribute Value remains empty, since the time stamp is the value of such variables. When creating the *Chronicle file*, the time stamp was also left empty for such variables. The program assigns the time stamp to the Value field of such variables. Whereas initially these values are stored as text, in later steps of the program all variables containing dates are formatted as dates in STATA. After adjusting Values of these variable Types, the program merges the field Transition to the extraction, obtained from the *Variable setup file*. This information is used by later steps of the program to correctly rectangularize each Type included in the extraction.

Some variable Types may have date collisions for an individual if there is more than one change in the Value of such Type on the same date. For individuals and Types where there are date collisions, a fraction of a day is added to the date of change if a value was included in the DayFrac field. For example, a variable indicating whether the previously born child is still alive (Prev_child_indicator) could assume three different Values: 1 (alive), 2 (dead and less than two years elapsed from the previous birth) or 3 (dead and more than two years elapsed from the previous birth). If a child is born and dies on the same day, the child's mother would have two different Values for the Type Prev_child_indicator on such date: 1 and 2. For this variable, the DayFrac could be left empty for the row that corresponds to Value 1 and 0.01 could be indicated in the row corresponding to Value 2. The program adds the DayFrac to the Date. In this example the Value 1 is the first row and Value 2 is the second row in chronological order. To adjust the date collisions the program adds 0.01 to the date October 7, 1855 of the row corresponding to Value 2, which means that the Value 1 would appear first and Value 2 second.

After adjusting date collisions, the **EFC** checks that all Values of each combination of Id_I, Type and

adjusted date are unique. An error message is given if problems are encountered for some variables, which are listed by the program. Such errors have to be fixed in the *Chronicle file* and *Variable setup file*, after which the **EFC** can be restarted, giving the corrected files during input.

PART 3 CHECKING THAT THE CHRONICLE AND VARIABLE SETUP FILES CONTAIN THE SAME TYPES

Part 3 of the **EFC** controls whether the *Variable setup file* and *Chronicle file* contain the same variable Types. If there are differences, a list of the Types not found in either of the files is given and the program is stopped. These inconsistencies have to be fixed in the *Chronicle file* and *Variable setup file*, after which the **EFC** can be restarted, giving the corrected files during input.

PART 4 RECTANGULARIZATION OF TIME-VARYING VARIABLES

Part 4 of the **EFC** rectangularizes time-varying variables, in other words all Types which change Value at the start of the spell and which have a value of Transition equal to Start (e.g. Civil_status). Each variable Type is transformed into a column heading, each different combination of Id_I and date is transformed into a row heading, and each Value is assigned to the corresponding cell of each specific Id_I, date and Type. The date variable is renamed date1, which allows the program to later merge these variables to the beginning of the spell. For very large extractions this step may take a long time to complete. The output of this step is saved with the name *Covariates_time_varying.dta*, which is a table containing one column per each time-varying variable included in the extraction. In part 7 of the **EFC** the file is merged to the start date of each spell (date1).

PART 5 RECTANGULARIZATION OF TIME-INVARIANT VARIABLES

Part 5 of the **EFC** rectangularizes time-invariant variables, in other words Types which are constant across all spells and which have a value of Transition equal to Invariant (e.g. sex). Each variable Type is transformed into a column heading, each Id_I is transformed into a row heading, and each Value is assigned to the corresponding cell of each specific Id_I and Type. The output of this step is saved with the name *Covariates_time_invariant.dta*, which is a table containing one column per each time-invariant variable included in the extraction. In part 7 of the **EFC** this file is merged to the spells, assigning the same value to all rows of the individual.

PART 6 RECTANGULARIZATION OF EVENTS

Part 6 of the **EFC** is aimed at rectangularizing events. Events are the variables of interest of the specific study and they can take place at the end dates of spells (e.g. death). All the rows of the *Chronicle file* which have a value of Transition equal to End are selected in this part of the program. The **EFC** transforms each event Type into a column heading, each different combination of Id_I and date into a row heading, and assigns the Value for each combination of Id_I, Type and date to the corresponding cells. The date variable is renamed date2. The output of this step is saved with the name *Events_end_dates.dta*. In part 7 of the **EFC** this file is merged to the end date of each spell.

PART 7 CONSTRUCTION OF SPELLS

The purpose of part 7 of the **EFC** is to create spells of consecutive dates. Spells are periods of time when time-varying variables remain constant and at the end of which events can take place. Each row in the table of spells begins with a date of change in any variable or event and ends with the next date of change in any variable or event.

To construct spells, the **EFC** first selects all unique dates of change from the *Chronicle file* and sorts them in ascending chronological order. Spells are created by defining as the start date of the spell (date1) the date of a row, and as end date of the spell (date2) the date of the succeeding row. Spells that have a date1 but no date2 (i.e. those that begin on the last exit date) are dropped.

After creating spells, the file containing time-varying variables saved in Part 4 (*Covariates_time_varying.dta*) is merged to the date1 of the spell. The file containing events saved in Part 6 (*Events_end_dates.dta*) is merged to the date2 of the spell, since events occur at the end dates of spells. Time-invariant variables saved in Part 5 (*Covariates_time_invariant.dta*) are merged, assigning the same value to all rows of the same individual. At the end of this part of the **EFC**, the file *PreEpisodes_file.dta* is saved. Some of the previously saved temporary files are erased.

PART 8 *FORMATTING OF THE EPISODES FILE*
Part 8 of the **EFC** formats the episodes table based on the information specified in the *Variable setup file*. All Values of Types included in the *Chronicle file* are stored as text. Variables containing date Values are reformatted as dates. Variables or events are converted by the program into numerical fields if they only contain numerical Values.

The *Chronicle file* given during input to the **EFC** contains all dates of change in every time-varying variable included in the extraction. In many cases such information is not only valid on the date of change in the Value, but it is valid until the next date of change. Missing cells of variables where the field Duration was set to Continuous in the *Variable setup file* are filled down by the program by copying the Value of the row above of the same individual (i.e. the date that precedes chronologically). Values are not filled down for events (Transition = End) or for those Types where the field Duration was set to Instant in the *Variable setup file*.

The program **append contextual variables** temporarily assigned the Value "NoValue" on the date of entry of an individual into a context if there were no Values of such variable declared on or before such date of entry. All Values "NoValue" are replaced by empty cells. Next, for all numerical time-varying and time-constant variables where the field Duration of the *Variable setup file* was set to Continuous, any remaining missing Values of the variable are replaced with -1.

For categorical numerical variables where Value labels were specified using an external file, the label is added to the Values of that variable.

PART 9 *DROPPING SPELLS WHEN THE INDIVIDUAL IS NOT AT RISK*
The final step of the **EFC** drops all spells when the individual is not at risk of experiencing the event of interest. The periods at risk are defined by the variable specified by the researcher in the program's syntax. This variable is only used for selecting spells when the individual is at risk, and is therefore dropped from the final episodes table.

The output of the program is saved with the name "Episodes_file_*time_string*". The time string contains the date and time when the final file is saved. All remaining temporary files used by the program and saved in the steps above are erased from the working directory.

Syntax
EpisodesFileCreator [using *labelFile*], atrisk(*atRiskVariable*)

Specifications:

labelFile: external file containing labels for variable Values (optional).

atRiskVariable: name of the variable included in the Chronicle file which defines the period when individuals are at risk. The variable should have value 1 on the date when the individual starts being at risk, and 0 on the date when the individual stops being at risk. Gaps can also be defined in this same way (0 when the individual exits the database and 1 when he/she returns).

10 EXAMPLE OF THE USE OF THE PROGRAMS PRESENTED IN THIS ARTICLE

This section presents an example that uses the programs discussed in this work to construct an episodes file for a study on mortality, based on the data included in the demo database. The example described is the same as that included in the file *UseIDSprograms_example.do*, which is attached to this article.

Before running any of the programs the Excel tables of the demo database are imported to STATA using the STATA command import, and each table is saved as a STATA data file. Using the program **extended IDS table maker**, the INDIVIDUAL_EXT, CONTEXT_EXT, Chronicle and Variable setup tables are created.

IDSextended , chronicle(1) varSetup(1) individual(1) context(1)

The tables created by the program are empty. The demo database includes a table containing extended individual level variables (Civil_status, At_risk_mortality, At_risk_fertility, Prev_child_indicator and Child_birth) and a table with their corresponding metadata. This information is appended to the INDIVIDUAL_EXT and METADATA tables using the program **import Data**. The demo database also includes contextual-level extended variables (Head_occupation, Number_of_servants) with the corresponding metadata. These variables are appended to CONTEXT_EXT and METADATA tables also using the program **import Data**. After running these steps, variables are available in all of the IDS and the EIDS tables.

importData INDI_EXT_import IND_MET_Import INDIVIDUAL_EXT

importData CONT_EXT_import.dta CON_MET_Import CONTEXT_EXT

The program **household size** is next used to calculate the size of each household. By running the program this time-varying variable is appended to the table CONTEXT_EXT and information on this variable is added to the METADATA table. As described earlier, changes in household size are linked to each household identifier.

householdSize

The IDS and the EIDS tables now contain all of the variables that are needed for the analysis. The program **select type** is used to create Excel files which allow the researcher to select information from the tables INDIVIDUAL, INDIVIDUAL_EXT and CONTEXT_EXT.

selectType, contextExt(1) individualExt(1) individual(1)

Variables for analysis are selected by opening the file INDIVIDUAL_SELECT.xls and inserting the value 1 in the field Select for the variables Birth_date, Birth_location, Death, and Sex. Birth_date, Birth_location and Sex are time-invariant variables, while Death is the event of interest of the study. The field Duration is set to Instant for the variable Death (since it is an event only valid on a specific date) and to Continuous for the other variables. The Excel file *INDIVIDUAL_EXT_SELECT.xls* is opened next, and the value 1 is inserted in the field Select for the variable At_risk_mortality, setting the field Duration to Continuous. This variable defines the period when individuals are at risk of experiencing death, which is from the date of birth or in-migration until the date of death or out-migration. In the file *CONTEXT_EXT_SELECT.xls*, the value 1 is inserted in the field Select of the variables Head_occupation, Household_size and Number_of_servants. The field Duration is set to Continuous for Hosuehold_size and Number_of_servants. The option is set to Instant for Head_occupation, since exact dates of change for this variable are not known and therefore different specifications can be tested at the time of making the analysis. Each Excel file is saved before closing it.

Information from the IDS and the EIDS tables that was selected through the Excel files is appended to the *Chronicle* and *Variable setup* files using the programs **append individual variables** and **append contextual variables**. An example of the *Chronicle file* and a *Variable setup file* produced from the demo database with the selections described above is shown in Table 4 and Table 5. These tables include the time-invariant variables Birth_date, Birth_location and Sex, the time-varying variables Head_occupation, Number_of_servants and Household_size and the event Death. It also includes the variable At_risk_mortality, which defines when the individuals were at risk of experiencing death, the event of interest of the study.

appendIndivVar, table(INDIVIDUAL)

appendIndivVar, table(INDIVIDUAL_EXT)

appendContextVar, table(CONTEXT_EXT)

Table 4 *Example of a Chronicle file for a mortality study*

Id_I	Type	Value	Day	Month	Year	DayFrac
1148964	Birth_date	1804-11-8	8	11	1804	
1148964	At_risk_mortality	1	11	4	1825	
1148964	Head_occupation	Farmer	11	4	1825	
1148964	Household_size	2	11	4	1825	
1148964	Number_of_servants	NoValue	11	4	1825	
1148964	Household_size	3	18	11	1828	
1148964	Household_size	2	15	9	1836	
1148964	Household_size	3	17	11	1836	
1148964	Number_of_servants	1	17	11	1836	
1148964	Household_size	4	10	8	1837	
1148964	Household_size	3	15	4	1838	
1148964	Household_size	2	8	6	1849	
1148964	Household_size	1	12	1	1852	
1148964	Number_of_servants	0	12	1	1852	
1148964	At_risk_mortality	0	8	11	1855	
1148964	Death		8	11	1855	
1148964	Birth_location	Kävlinge				
1148964	Sex	Male				
1237852	Birth_date	1807-4-12	12	4	1807	
1237852	At_risk_mortality	1	11	4	1825	
1237852	Head_occupation	Farmer	11	4	1825	
1237852	Household_size	2	11	4	1825	
1237852	Number_of_servants	NoValue	11	4	1825	
1237852	Household_size	3	18	11	1828	
1237852	At_risk_mortality	0	15	9	1836	
1237852	At_risk_mortality	1	10	8	1837	
1237852	Household_size	4	10	8	1837	
1237852	Number_of_servants	1	10	8	1837	
1237852	At_risk_mortality	0	15	4	1838	
1237852	Death		15	4	1838	
1237852	Birth_location	Hög				
1237852	Sex	Female				
1378563	At_risk_mortality	1	18	2	1853	
1378563	Birth_date	1853-2-18	18	2	1853	
1378563	Head_occupation	Farmer	18	2	1853	
1378563	Household_size	4	18	2	1853	
1378563	Number_of_servants	1	18	2	1853	
1378563	At_risk_mortality	0	1	12	1874	
1378563	Birth_location	Lund				
1378563	Sex	Female				
1479856	Birth_date	1831-12-15	15	12	1831	
1479856	At_risk_mortality	1	1	2	1852	
1479856	Household_size	3	1	2	1852	
1479856	Number_of_servants	1	1	2	1852	
1479856	Head_occupation	Farmer	18	2	1853	
1479856	Household_size	4	18	2	1853	
1479856	Household_size	3	1	12	1874	

Table 4 continued on next page

Id_I	Type	Value	Day	Month	Year	DayFrac
1479856	Head_occupation	-1	3	6	1878	
1479856	Household_size	2	3	6	1878	
1479856	At_risk_mortality	0	14	11	1881	
1479856	Birth_location	Lund				
1479856	Sex	Male				
1548468	At_risk_mortality	1	7	10	1855	
1548468	At_risk_mortality	0	7	10	1855	0.01
1548468	Birth_date	1855-10-7	7	10	1855	
1548468	Death		7	10	1855	
1548468	Household_size	4	7	10	1855	
1548468	Number_of_servants	1	7	10	1855	
1548468	Birth_location	Lund				
1548468	Sex	Female				
1567526	Birth_date	1821-7-26	26	7	1821	
1567526	At_risk_mortality	1	16	9	1851	
1567526	Head_occupation	Farmhand	16	9	1851	
1567526	Household_size	2	16	9	1851	
1567526	Number_of_servants	NoValue	16	9	1851	
1567526	Household_size	3	1	2	1852	
1567526	Number_of_servants	1	1	2	1852	
1567526	Head_occupation	Farmer	18	2	1853	
1567526	Household_size	4	18	2	1853	
1567526	Household_size	3	1	12	1874	
1567526	Head_occupation	-1	3	6	1878	
1567526	Household_size	2	3	6	1878	
1567526	Household_size	1	14	11	1881	
1567526	Number_of_servants	0	14	11	1881	
1567526	At_risk_mortality	0	4	8	1885	
1567526	Death		4	8	1885	
1567526	Birth_location	Kävlinge				
1567526	Sex	Female				
1897563	Birth_date	1819-8-5	5	8	1819	
1897563	At_risk_mortality	1	17	11	1836	
1897563	Household_size	3	17	11	1836	
1897563	Number_of_servants	1	17	11	1836	
1897563	Household_size	4	10	8	1837	
1897563	Household_size	3	15	4	1838	
1897563	Household_size	2	8	6	1849	
1897563	At_risk_mortality	0	12	1	1852	
1897563	Birth_location	Malmö				
1897563	Sex	Male				
1945568	At_risk_mortality	1	18	11	1828	
1945568	Birth_date	1828-11-18	18	11	1828	
1945568	Household_size	3	18	11	1828	
1945568	Number_of_servants	NoValue	18	11	1828	
1945568	Household_size	2	15	9	1836	
1945568	Household_size	3	17	11	1836	
1945568	Number_of_servants	1	17	11	1836	

Table 4 continued on next page

Id_I	Type	Value	Day	Month	Year	DayFrac
1945568	Household_size	4	10	8	1837	
1945568	Household_size	3	15	4	1838	
1945568	At_risk_mortality	0	8	6	1849	
1945568	At_risk_mortality	1	16	9	1851	
1945568	Head_occupation	Farmhand	16	9	1851	
1945568	Household_size	2	16	9	1851	
1945568	Number_of_servants	NoValue	16	9	1851	
1945568	Household_size	3	1	2	1852	
1945568	Number_of_servants	1	1	2	1852	
1945568	Head_occupation	Farmer	18	2	1853	
1945568	Household_size	4	18	2	1853	
1945568	Household_size	3	1	12	1874	
1945568	At_risk_mortality	0	3	6	1878	
1945568	Death		3	6	1878	
1945568	Birth_location	Malmö				
1945568	Sex	Male				

Table 5 *Example of a Variable setup file for a mortality study (linked to Table 4)*

Type	Duration	Transition
At_risk_mortality	Continuous	Start
Birth_date	Continuous	Invariant
Birth_location	Continuous	Invariant
Death	Continuous	End
Head_occupation	Instant	Start
Household_size	Continuous	Start
Number_of_servants	Continuous	Start
Sex	Continuous	Invariant

The program **EFC** is used to create an episodes table based on the *Chronicle file* and the *Variable set-up file* presented in Tables 4 and 5. At_risk_mortality is specified in the syntax of the program as the variable defining the period at risk.

EpisodesFileCreator, atrisk(At_risk_mortality)

Table 6 shows the episodes table that is produced during output from the program **EFC** with the files and specifications described above. The table contains the individual identifier (Id_I), the start and end dates of each spell (date1 and date2), the time-varying variables Head_occupation, Household_size, Number_of_servants, the time-invariant variables Birth_location, Sex and Birth_date and the event Death. All variable Values have been copied down to dates succeeding changes in these variables (i.e. missing cells have been filled in), except for Head_occupation, based on the specifications described above. In the table, all numerical variables are formatted as numbers and all date variables are formatted as dates. Only spells during which the individual was at risk of experiencing the event of interest are included, which in this case corresponds to all the spells when the individuals were present in the dataset (i.e. from birth or in-migration until death or out-migration).

If the demo database would have included real and more extended data, the episodes table presented would have been ready for statistical analysis. It could be used, for example, to study the impact of household structure, sex and birth location on mortality by using Cox proportional hazards models. Based on the date1, date2 and Birth_date, the dataset could be split to estimate models for different age groups. Different specifications of Head_occupation could also be tested in the analysis, for example considering occupations valid until the next date of change, or until a certain amount of time between consecutive dates of declaration in such variable.

Table 6 *Example of an episodes table for a mortality study (constructed from Table 4 and 5)*

Id_I	date1	date2	Head_occupation	House-hold_size	Number_of_servants	Birth_location	Sex	Birth_date	Death
1148964	11apr1825	18nov1828	Farmer	2	-1	Kävlinge	Male	08nov1804	
1148964	18nov1828	15sep1836		3	-1	Kävlinge	Male	08nov1804	
1148964	15sep1836	17nov1836		2	-1	Kävlinge	Male	08nov1804	
1148964	17nov1836	10aug1837		3	1	Kävlinge	Male	08nov1804	
1148964	10aug1837	15apr1838		4	1	Kävlinge	Male	08nov1804	
1148964	15apr1838	08jun1849		3	1	Kävlinge	Male	08nov1804	
1148964	08jun1849	12jan1852		2	1	Kävlinge	Male	08nov1804	
1148964	12jan1852	08nov1855		1	0	Kävlinge	Male	08nov1804	1
1237852	11apr1825	18nov1828	Farmer	2	-1	Hög	Female	12apr1807	
1237852	18nov1828	15sep1836		3	-1	Hög	Female	12apr1807	
1237852	10aug1837	15apr1838		4	1	Hög	Female	12apr1807	1
1378563	18feb1853	01dec1874	Farmer	4	1	Lund	Female	18feb1853	
1479856	01feb1852	18feb1853		3	1	Lund	Male	15dec1831	
1479856	18feb1853	01dec1874	Farmer	4	1	Lund	Male	15dec1831	
1479856	01dec1874	03jun1878		3	1	Lund	Male	15dec1831	
1479856	03jun1878	14nov1881	-1	2	1	Lund	Male	15dec1831	
1548468	07oct1855	07oct1855		4	1	Lund	Female	07oct1855	1
1567526	16sep1851	01feb1852	Farmhand	2	-1	Kävlinge	Female	26jul1821	
1567526	01feb1852	18feb1853		3	1	Kävlinge	Female	26jul1821	
1567526	18feb1853	01dec1874	Farmer	4	1	Kävlinge	Female	26jul1821	
1567526	01dec1874	03jun1878		3	1	Kävlinge	Female	26jul1821	
1567526	03jun1878	14nov1881	-1	2	1	Kävlinge	Female	26jul1821	
1567526	14nov1881	04aug1885		1	0	Kävlinge	Female	26jul1821	1
1897563	17nov1836	10aug1837		3	1	Malmö	Male	05aug1819	
1897563	10aug1837	15apr1838		4	1	Malmö	Male	05aug1819	
1897563	15apr1838	08jun1849		3	1	Malmö	Male	05aug1819	
1897563	08jun1849	12jan1852		2	1	Malmö	Male	05aug1819	
1945568	18nov1828	15sep1836		3	-1	Malmö	Male	18nov1828	
1945568	15sep1836	17nov1836		2	-1	Malmö	Male	18nov1828	
1945568	17nov1836	10aug1837		3	1	Malmö	Male	18nov1828	
1945568	10aug1837	15apr1838		4	1	Malmö	Male	18nov1828	
1945568	15apr1838	08jun1849		3	1	Malmö	Male	18nov1828	
1945568	16sep1851	01feb1852	Farmhand	2	-1	Malmö	Male	18nov1828	
1945568	01feb1852	18feb1853		3	1	Malmö	Male	18nov1828	
1945568	18feb1853	01dec1874	Farmer	4	1	Malmö	Male	18nov1828	
1945568	01dec1874	03jun1878		3	1	Malmö	Male	18nov1828	1

11 CONCLUSIONS

This article presented seven programs which can be used to construct a file for analysis from data stored in the IDS. The programs presented allow the creation of the EIDS tables and append imported data to them, to construct a variable to indicate household size, to select variables for analysis from those included in the IDS and the EIDS tables, to produce a data extraction from them and to transform this data extraction into a rectangular episodes file that is ready for statistical analysis. The specifications and syntax of these programs was presented in detail in this article and an example of how to use them was also given.

The use of the programs presented in this work has several advantages. They allow the user to create datasets for analysis by selecting information stored in the IDS and the EIDS tables without the need of conducting any further re-elaborations, therefore facilitating the work of database administrators and researchers. By using these programs researchers can directly select the variables that are required for their analysis, and they can also define how to format these selected variables. The programs are very easy to use, which allows students and researchers with any level of experience in data management to obtain a dataset for analysis without conducting any further programming. Since the programs are developed to be used with the IDS structure and they use generic code, they can be used for any database or combination of databases produced from data obtained from family reconstitutions or population registers. Moreover, any one of these programs can be used independently and in combination with other open access or local programs, as long as the specifications outlined in this article and in Quaranta (2015) are met. The rectangularized file given during output by the programs is saved as a STATA data file, but it can also be easily exported to be used for analysis with other statistical packages.

Over the past decades the field of historical demography has widely expanded, partly thanks to the availability of digitized micro-level longitudinal databases and to the development of new and highly advanced statistical techniques. Although today there are numerous programs that allow users to conduct advanced statistical modelling using simple commands, not many programs have been developed to simplify the complexity of the data management that is needed to create datasets suitable for conducting such statistical analyses. The IDS was developed in order to solve some of the challenges related to using longitudinal demographic data, in particular by providing a structure for storing and sharing data and which can simplify the development of open-access software. This work and the article *Using the Intermediate Data Structure (IDS) to Construct Files for Statistical Analysis* have discussed different concepts and programs that can be used to construct a dataset for analysis from data stored in IDS. These concepts and programs even further expand the scope and flexibility of the IDS and the use and sharing of historical demographic data for research.

ACKNOWLEDGEMENTS

I would like to thank George Alter, Clas Andersson and Kees Mandemakers for their advice and help and Anders Brändström, Hilde Leikny Sommerseth, Annika Westberg and two anonymous reviewers for their comments. I am also grateful for the travel funding received from the EHPS-net and for administrative assistance from Marja Koster and Paul Puschmann.

REFERENCES

Alter, G. & Mandemakers, K. (2014). The Intermediate Data Structure (IDS) for longitudinal historical microdata, version 4. *Historical Life Course Studies,* 1, 1-26.

Alter, G., Mandemakers, K. & Gutmann, M. (2009). Defining and distributing longitudinal historical data in a general way through an intermediate structure. *Historical Social Research,* 34(3), 78-114.

Cox, D. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society, Series B (Methodological)*, 34(2), 187-220.

Quaranta, L. (2015). Using the intermediate data structure (IDS) to construct files for statistical analysis. *Historical Life Course Studies,* 2, 86-107.

Stead, W., Hammond, W. & Straube, M. (1982). A chartless record - is it adequate? In: *Proceedings of the annual symposium on computer application in medical care* (p. 89-94). *American Medical Informatics Association.*
DOI: 10.1007/BF00995117

Therneau, T. M. & Grambsch, P. M. (2000). *Modeling survival data: Extending the Cox model*. New York: Springer-Verlag.