

Linking 170 Million Records With SAS

Contents

Linking Big (ish) Government Datasets Using SAS and A Few Other Friends	2
1. Requirement	2
2. The Overall Picture.....	3
IR Data.....	3
DWP Data.....	3
IT Facilities.....	3
3. Linking The Data.....	4
Data Available	4
Linking Process – Placing DWP Identifier on IR Data	5
Linking Process – Creating a Full Benefit / Employment History	6
4. Security	6
5. Analysis	7
6. The “Friendly Front End”	8
Work and Pensions Longitudinal Study (WPLS) Interface.....	9
Why Do We Need An Interface?.....	9
An Interface Already in Production.....	10
Principles of The WPLS Interface.....	10
Template Code.....	10
Self Development	10
Flexible Use.....	11
Possibility of Removing Some Functionality	11
A Tour of The Interface.....	13
SAS Windows	13
User Profiles and Autoexecs.....	13
Selection Index	14
The Training Slides and Dataset index.....	16
Summary.....	16
Future Development	16
Contact Details	17
Annex A – Coverage of Benefits and Programmes.....	18
Annex B - Soundex	19
Annex C – SAS Code for Fuzzy Matching Process	20

Linking Big (ish) Government Datasets Using SAS and A Few Other Friends

Carol Calvert
Statistician



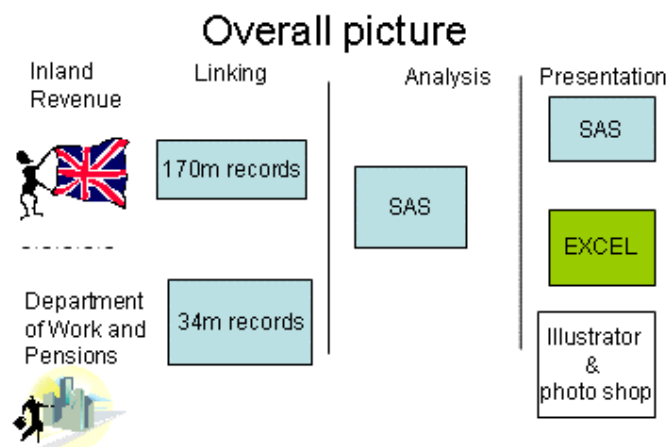
What happens when working
age people leave benefit?

The Work and Pensions
Longitudinal Study



1. Requirement

- 1) To link the individual records in the Department of Work and Pensions (DWP) benefits / programme database with employment data from Inland Revenue (IR). The DWP database has around 34 million records <10Giga Bytes> covering details from 1999 to date; the IR data has 170 million records <43Giga Bytes> covering April 1998 to date.
- 2) To provide a source of linked data for use by analysis teams to evaluate, in terms of take up and retention in employment, leavers from DWP programmes such as New Deal.



2. The Overall Picture

IR Data

The data gives employment start and employment end data and is based on P45/P56 information sent to Inland Revenue by employers. The corresponding information for the self-employed will be received from May 2004. The data we currently have covers the period 6th April 1998 to date and is approximately 170 million records <43 Giga Bytes>. An individual may have several records as one record relates to each change of information about each employment and this may, for example, involve changes in payroll number, in employment date, or tax office.

The data is supplied from the IR data warehouse and the back data was supplied as 10 files each of 4.3Giga Bytes as pipe delimited files. From May monthly 'change' data will be supplied in a similar manner.

DWP Data

This covers all programs like New Deal and all benefits like Income Support, JobSeeker's Allowance and Incapacity Benefit. The index database is held in SAS and has multiple records per person; one per spell per benefit /programme. It contains 34 million records and includes a unique identifier which we use to identify the full details held in separate databases for each benefit / programme (see Annex A). The index data has around 34 million records <10 Giga Bytes>. Information is held back to 1999.

IT Facilities

The data is stored on a single 8 processor UNIX server running HP Tru64 Unix at our Newcastle site. The data is stored as SAS files and the linking programs are run remotely from our Sheffield site using SAS 8.2 TS Level 02 M0.

Fuzzy match

- NINO
- Surname
- Forename
- Postcode
- Date of birth
- Gender

- `filt5=compress(soundex(surname)||
substr(forename,1,1)|| sexir||
scan(pcodeir,1)||substr(scan(pcodeir,2),1,1)||
date_of_birth);`

3. Linking The Data

Data Available

The IR data and the DWP data contain six common pieces of information:

- National Insurance number - NINO
- Surname
- Initial
- Date of Birth - DOB
- Gender
- Address.

NINO is potentially a unique identifier but there are a number of difficulties associated with it. Therefore both the IR and DWP have independently developed their own unique identifiers but, unfortunately, the two identifiers are not the same. The choice was made, in developing the linking between the two data sets to place the DWP identifier on the IR data. This enabled maximum flexibility within DWP as all our data sets carry the unique number adopted by the Department.

Surname is a well completed field and the SOUNDSEX algorithm within SAS is used to help overcome minor differences in spelling. The algorithm is an international standard which codes similar sounding words- see annex B.

Some, but not all, of the datasets that the DWP index file calls upon contain full first and second names. **First initial** was therefore the only element that could be consistently accessed across all data sets.

Date of birth and **gender** were consistently available. The IR data is read into SAS and the DWP data is already in SAS and hence all dates are stored as SAS dates.

Address data is notoriously difficult to deal with given the variety of ways elements are recorded. The postcode therefore offered a better option for matching. The IR address fields are effectively the address when the individual started with the employer. The address is unlikely to be updated if the employee moved house whilst with the same employer. However most home moves are local and there is also a perception that most postcode errors involve the later characters. The **postcode sector**, rather than full postcode, has therefore been used for matching.

Database 1: off benefit / programme into work						
ORC identifier	Date started on benefit programme	Date left programme / benefit	Programme / benefit	Personal details age, gender, ethnicity, address etc.	Employer name, address and trade classification	employment start
6760006	17/12/1999	01/01/2001	New Deal Young People		Janson	02/01/2001
6760006	15/01/2000	01/01/2001	Joint Claims		Janson	02/01/2001
6760006	15/01/2000	01/01/2001	JSA		Janson	02/01/2001
Database 2: Benefit / programme spells database						
ORC identifier	Date started on benefit programme	Date left programme / benefit	Programme / benefit	Personal details age, gender, ethnicity, address etc.	Employer name, address and trade classification	employment start
6760101	14/07/2000	03/08/2000	JSA			
6760101	21/08/2000	15/10/2000	JSA			
6760101					Don't know	07/04/1998
6760101					Smiles law wing	18/07/1999
6760101					Smiles law wing	25/08/2000
6760101					Eumbrad	
6760101					Smiles law wing	10/01/2001
6760101					Eumbrad	08/08/2001
6760101					Eumbrad	27/02/2001
6760101					Eumbrad	21/03/2001
6760101					Eumbrad	17/05/2001
6760101					Eumbrad	29/09/2001
6760101					Castridazy	10/03/2002
6760006	17/12/1999	01/01/2001	New Deal Young People			
6760006	15/01/2000	01/01/2001	Joint Claims			
6760006	15/01/2000	01/01/2001	JSA			
6760006					La Sreca	10/08/1998
6760006					DIAMMY	06/04/2000
6760006					Largle Row	06/04/2000
6760006					Janson	02/01/2001

Dummy data

Linking Process – Placing DWP Identifier on IR Data

The first task was to identify corresponding records on the two data sources, using the six common variables, and then “tag” on the DWP identifier to the IR data. The IR data, including the DWP identifier, is then subsequently matched onto the details of those who had been or are current DWP clients. The result of this matching is a combined benefit and employment history for an individual. In many cases there will be no DWP identifier because the IR record is for an individual who is not currently, nor has been since 1999, a DWP client.

The data sets are too large (34 Million DWP records and 170 million IR records) to simply match all possible combinations of the common variables using a merge statement. A staged approach is therefore adopted. The data is matched on NINO as a first step. If the NINO on both data sources match the “quality” of the match is assessed using the other five common variables. If the NINO does not match then a composite of the five other common variables is generated for both remaining IR and DWP data. The new composite variable is checked to see if it matches across both data sets and if so, a match and the “quality” of the match are established. The process continues by generating new variables based on the five possible combinations of four common variables and then the 10 possible combinations of three variables (see annex C for example of code).

The “quality” of the match is based on both the number and “importance” of the matching variables. NINO is, despite its limitations, a good matching variable and hence if it does not match the overall quality of the match is diminished. The tendency for families to live locally and to sometimes share family first names means that surname, initial, gender and postcode sectors are often the same for different family members. In the absence of a NINO match the additional absence of date of birth further restricts the confidence that can be placed in the match and therefore the “quality” of the match is downgraded by one level (see table 1).

In cases where one of the matching variables is missing that particular variable is taken as a non match. The effect is to “down grade” the quality of the match by one level.

Table 1: “Quality” of matches

Matching variables	“Quality” of match	Number records
NINO and four or all five of DOB,gender,SOUNDEX(surname), initial,postcode sector	Green	6,810,000
NINO and three of DOB,gender,SOUNDEX(surname), initial,postcode sector	Amber	2,090,000
NINO and two of DOB,gender,SOUNDEX(surname), initial,postcode sector	Red	680,000
NINO and one or none of DOB,gender,SOUNDEX(surname), initial,postcode sector	Non Match	53,000
No NINO match but all of DOB,gender,SOUNDEX(surname), initial,postcode sector	Green/ Amber	30,000
No NINO match but DOB & three of gender,SOUNDEX(surname), initial,postcode sector	Amber / Red	860,00
No NINO or DOB match & all of: gender,SOUNDEX(surname), initial,postcode sector	Red/ Amber	3,320,000

Linking Process – Creating a Full Benefit / Employment History

Once the DWP unique identifier is placed on the IR data set then a match to the details on the DWP data set is made. As the data sets are large both data sets are divided into smaller files by month and year. Each month / year file is sorted and any repeat records on a particular benefit in that month / year for the individual removed with the record with the latest end date retained. This gives a unique record, per person per benefit type per month / year which is then linked to the appropriate IR data sets. There are six IR datasets in all, two for the preceding months, one for the same month, two for the succeeding months and one general file for the year where exact employment start dates are unknown.

The merge statement in a data step is used for the matching which is effectively a one-to-many match. In the future we will be receiving just “change” records from IR and our linking procedure will be simplified as the files will be much smaller. We are likely to use PROC SQL which will enable us to run a many-to-many match.

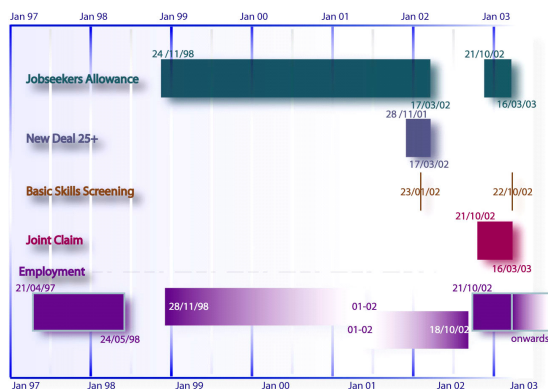
4. Security

In order to legally use the IR data the processes by which the data was stored, accessed and made available had to be in compliance with certain legislation. This legislation included the Data Protection Act, Human Rights Act, Social Security Act and the Government Statistical Code of Practice. The processes include;

- The establishment of an independent ethics committee

- Specific training and requirement of an approved business case for access to the data
- Storage of the data in read only datasets in secure specific directories
- Monitoring of access to the datasets
- Establishment of an Internet site detailing study and the use of the data.

Dummy data



5. Analysis

Three SAS datasets are made available for users and contain multiple records per individual. The first database links the benefit / programme records with employment records (21m records); the second contains the full benefit and employment records by individual (44 m records); and the third contains employment records for non DWP clients (84m records). In some cases analysts will directly use these SAS files. In other cases they will access the information via a ‘friendly front end’ – see section 6 below.

In some cases final analysis will be done in EXCEL and presentation may be in other packages. As the databases show the “history” of individuals one of the mechanisms we use is to present a ‘time line’ for an individual- currently using Illustrator or Photo Shop.

What’s possible

- Geography – national, by region, parliamentary constituencies, or any geography and combined with other information at the same level.
- Time – over years, months, before/ after introduction of initiatives, time on benefit, time in employment.
- Individual characteristics- age, gender, ethnicity etc
- Employer characteristics – location, size, trade classification

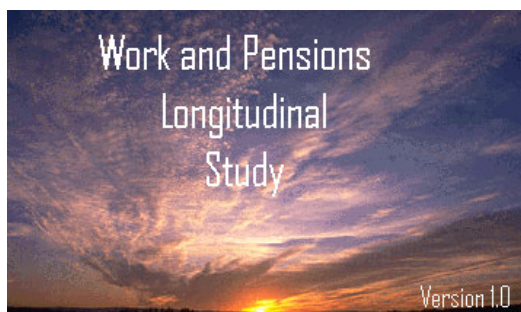
6. The “Friendly Front End”

An explanation of the “friendly front end” or Work and Pensions Longitudinal Study Interface will now follow.

Work and Pensions Longitudinal Study (WPLS)

Interface

Philip Rose
SAS Consultant



Why Do We Need An Interface?

1. Increased Workload

The work and pensions longitudinal study (WPLS) is a very large very powerful database. The potential of such a large pool of information has created interest from many analysts both within the Department of Work and Pensions and external agencies. This interest includes requests for ad-hoc or regular analysis reports as well as access to be able use the information for their selves.

2. Increased Dataset Size

Previous datasets were typically in the order of 10 million records. This increased 10 fold with the advent of the WPLS. To analyse such large datasets it became essential to ensure the most efficient programming techniques were used. For example an unnecessary sort, an ill defined subset or an incorrect join could mean the waste of hours or run time. Also it was very important that the user was guided as much as possible in how the various datasets were related to ensure the quality and reliability of the results.

3. SAS Unfamiliarity

The creation of the WPLS closely followed a decision to phase in the use of SAS as a replacement to resident packages such as SPSS. Therefore many analysts were relatively unfamiliar with the package they were to use.

Challenge

The challenge was to enable highly experienced SPSS users to achieve the same results in the same timescales (or better) and to continue the increase in productivity already being achieved by the organisation. All this whilst working with a new software package and much larger datasets.

An Interface Already in Production

This sort of challenge was not new to DWP. The recent switch to SAS coincided with a decision to use 100% data samples for routine analysis of benefit data. Previously analysis was mostly carried out on 5% samples of the national data. Therefore there had already been a 20 fold increase in the amount of data being analysed.

This challenge was addressed in an office in Newcastle where a point and click interface was developed in AF Frame. This interface could be seen as a wizard like application that worked similar albeit in a much more comprehensive way to the SAS import/export wizard or the where clause builder in the dataset viewers such as View Table. Operations including searching meta-data for specific information, viewing datasets, writing and submitting data steps and procedures, data analysis, report production, accessing a template syntax library could all be accomplished by users with minimal knowledge of SAS syntax. Standard SAS utilities such as “Quality Improvement” and “Graph-N-Go” could also be accessed via the interface.

The interface had been embraced by staff of wide ranging SAS expertise. Novices used it to its full potential whilst more competent SAS programmers used the interface almost as an online aid to help with those niggling pieces of syntax that you can never remember and normally have to resort to the manuals or on-line help.

There was one key difference between this pre-existing situation and the use of WPLS. Before the WPLS there were a multitude of datasets many requiring joining prior to analysis. Often a certain amount of standardisation was needed to allow these joins. With the WPLS the number of datasets was much reduced placing the emphasis on the need to subset the data prior to analysis. However the fundamentals of the interface in supplying an easy, standardised way to use SAS would still be valid for the WPLS.

It was decided to base the WPLS interface on this existing interface.

Principles of The WPLS Interface

Template Code

In producing and running SAS code the WPLS interface utilises a library of template programs. Depending on the task requested by the user the relevant template\’s are accessed. The variable pieces of syntax such as dataset names and variable names are requested from the user via a series of input boxes and selection lists. These are read into the template\’s.

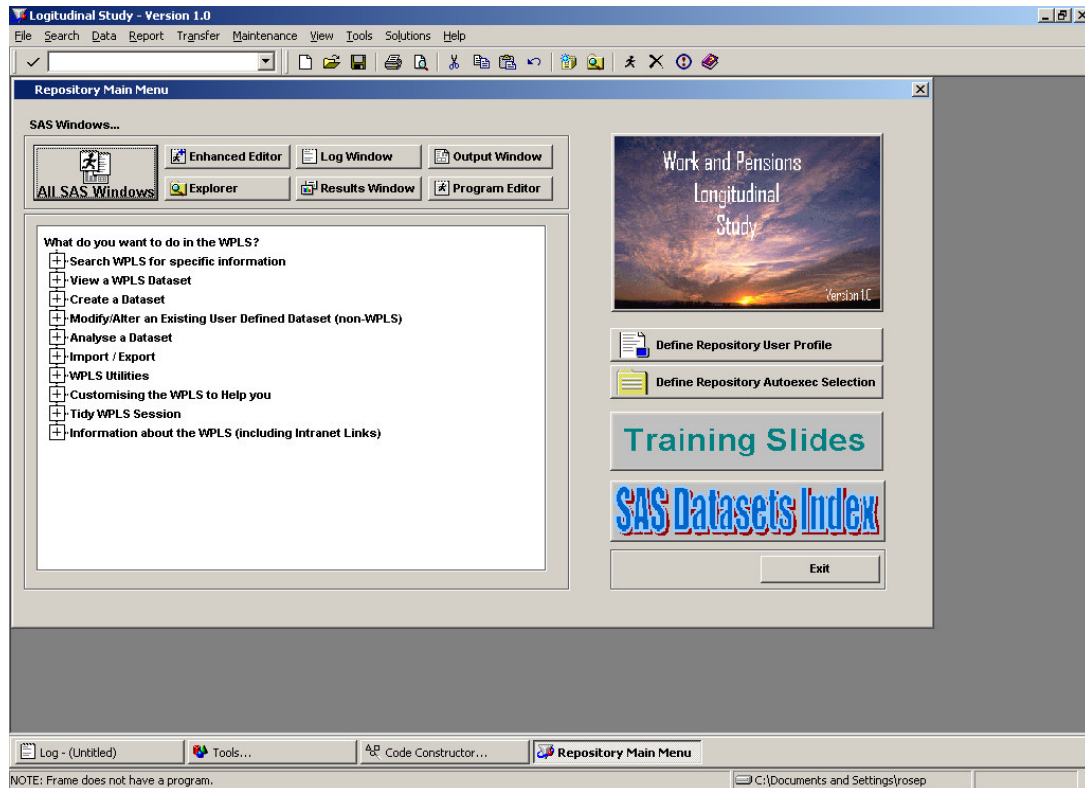
Self Development

The disadvantage of many applications of this sort is that whilst they help the productivity of the novice user they also create a barrier to the development of SAS expertise for those individuals. Users can often become experts in the application and never get chance to learn how to actually write code. A particular strength of this interface is that it can either

- perform the task without the user having to write or indeed see any code

- provide the user with the code for them to edit or integrate into a previously written program
 - provide template examples to be completed by the user thereby acting as a type of online help tool
- This flexibility encouraged staff development as well as increasing productivity.

Flexible Use



The way the features of an application are accessed by the user often governs whether the user likes the application and indeed whether they are willing to use it. Some users prefer push buttons, some prefer pull down menus whilst others prefer to select from an index of the functions similar to how they would use a book. All the methods were made available within our interface and will be discussed later. We feel that this was probably a key factor in the interfaces popularity and success.

Possibility of Removing Some Functionality

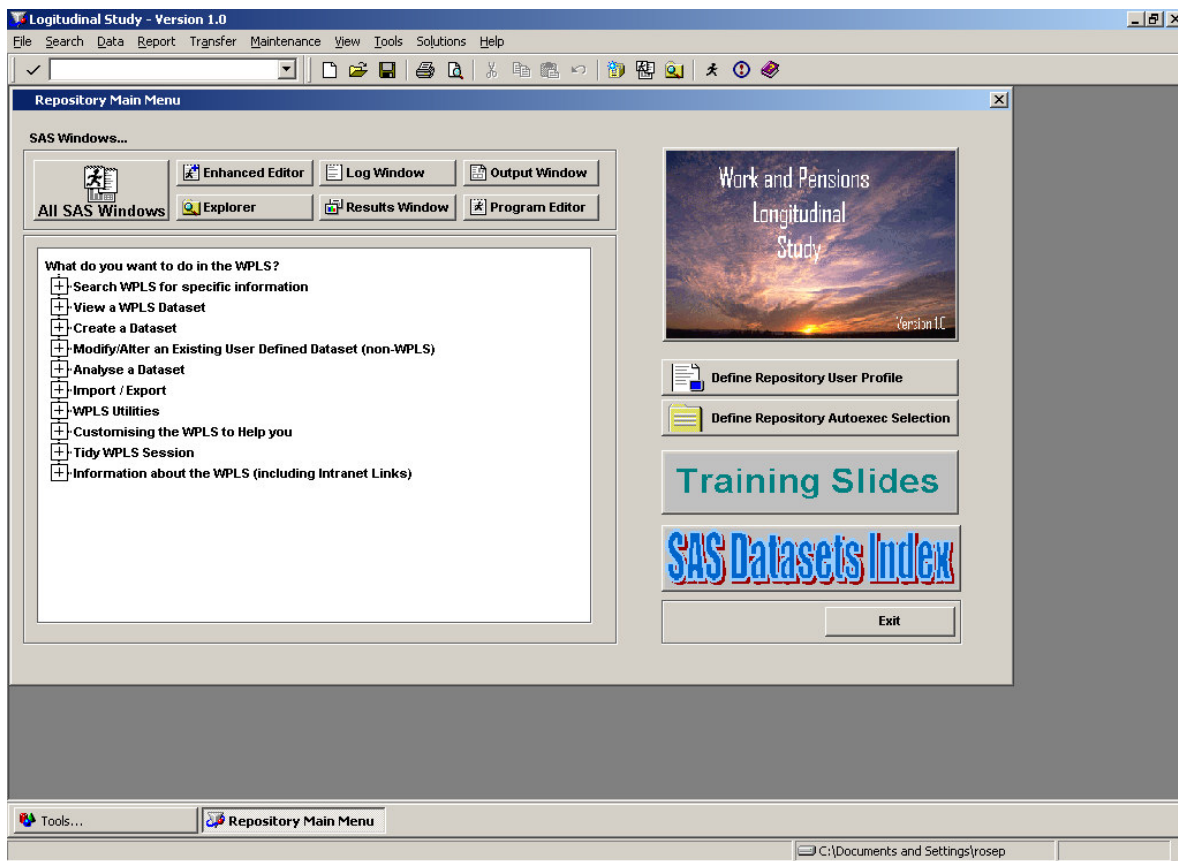
Initial intentions: It was initially decided as an added precaution to prevent the ability to copy or download the datasets in anyway. This would require removing access to the standard SAS windows in particular the enhanced and program editors and preventing the user for running any SAS code independently to the interface.

Problem: It became apparent that to limit the functionality of the interface so much would reduce the ability of the users to analyse the data efficiently. For example, during the course of analysis the users constantly created subsets and

copies of the data and manipulated the data via ad-hoc programming directly in the SAS editor. To remove this ability would decrease efficiency and turn the interface into a frustrating barrier rather than a useful tool. Secondly, such a limitation would effectively remove the ability of the interface to be used as a learning aid. Finally, to limit the functionality in such a way would not completely prevent misuse of the data. For example it would always be possible for an experienced SAS user to regain access to the editor and thus view or copy data illegally.

Solution: Therefore it was decided to leave the security features at the server level where viewing, copying, and downloading could be monitored. However one additional security feature would be advantageous and that was the capturing of every SAS log produced during the SAS session in which the interface ran. These logs could be scanned by security personnel for key words such as “copy”, “download” etc and the log flagged. Flagged logs could then be investigated further for evidence of misuse of the data (either accidental or malicious).

A Tour of The Interface

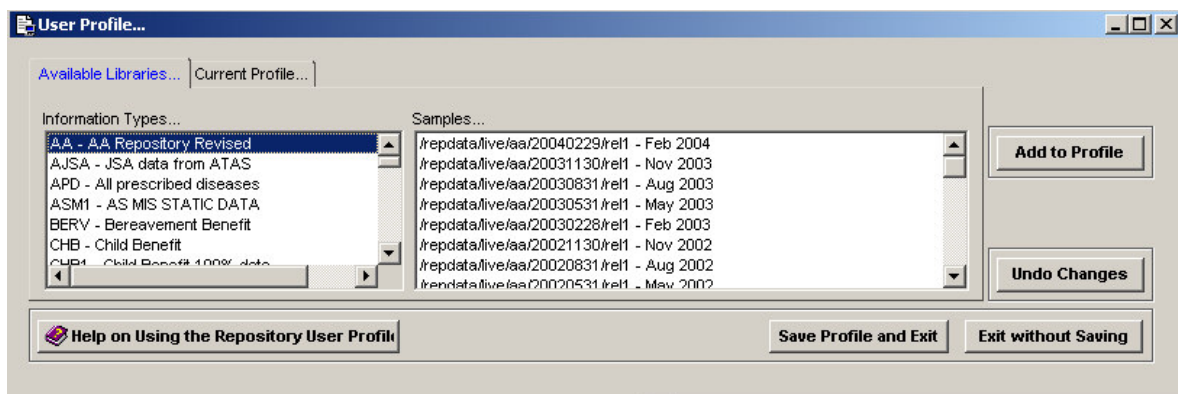


There are four basic features of the interface.

SAS Windows

Push buttons to open the standard SAS windows such as the enhanced editor, log and output windows etc.

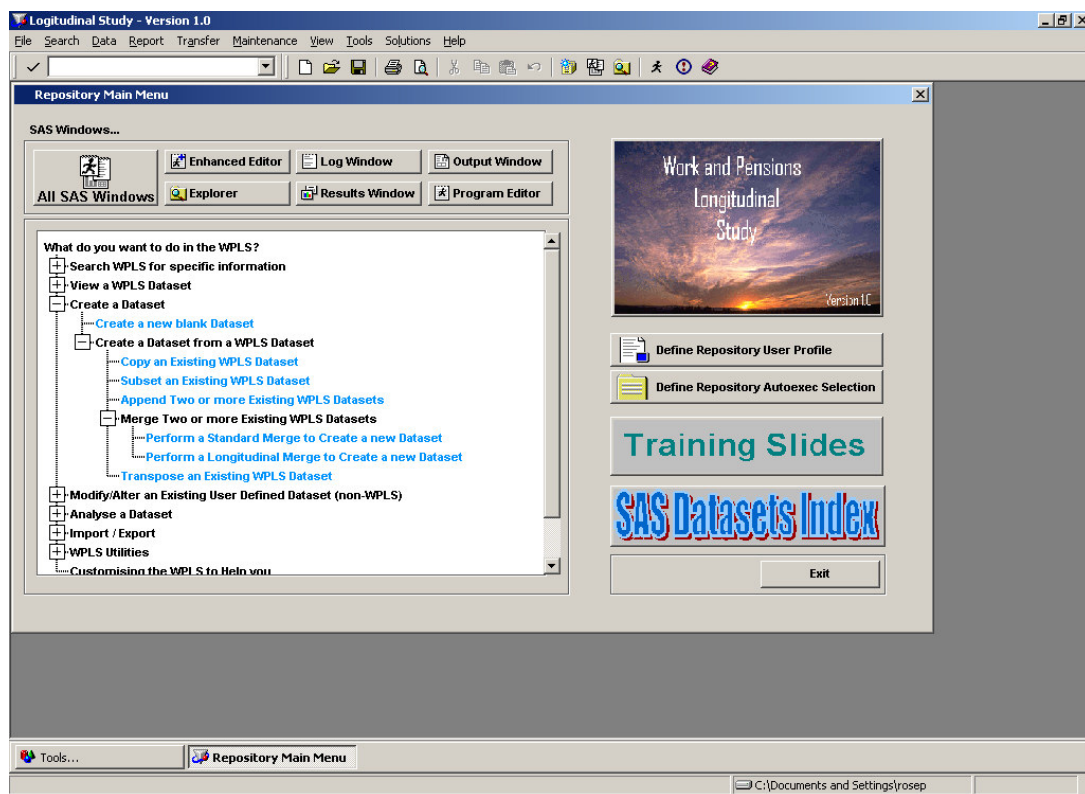
User Profiles and Autoexecs



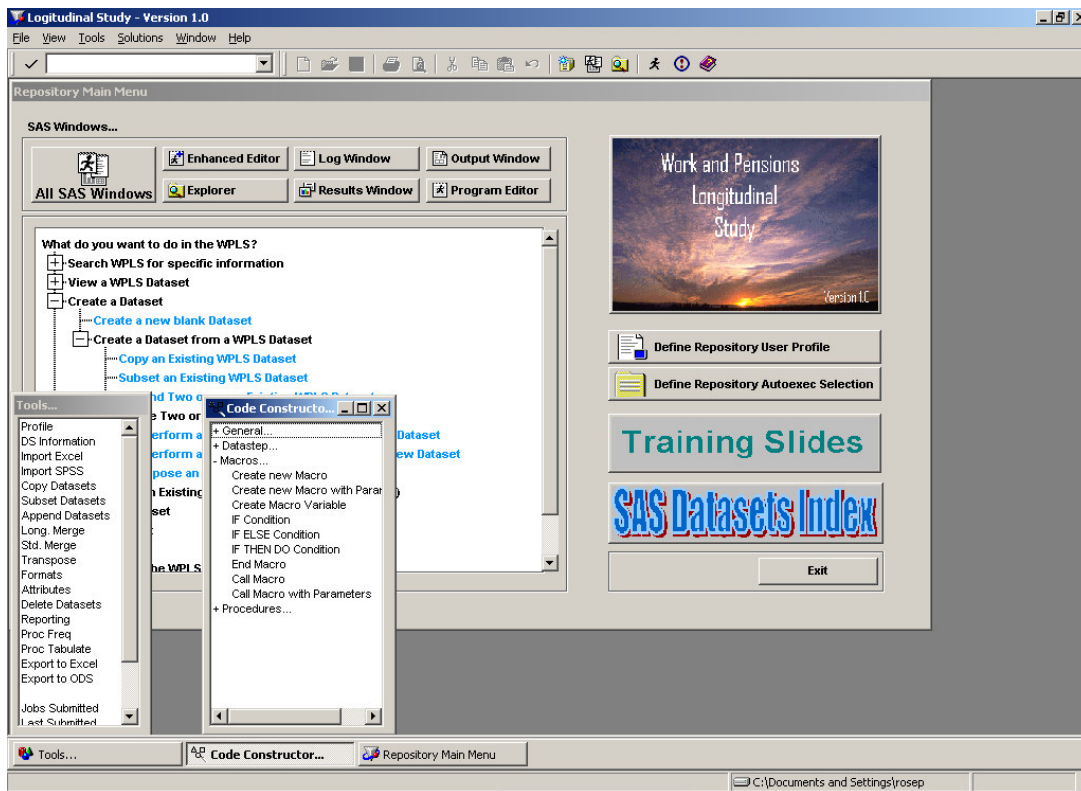
The provision of individual user profiles. Here library references to the various WPLS datasets can be defined and saved. These library references can then be automatically recreated rather than having to be manually set up every time the particular user accesses the interface. It was decided that to provide all libraries to all users would be a needless security hazard as users usually only require access to particular data.

As well as library references to the WPLS datasets the user can also define references to other areas such as their own allotted user areas on the server. These too can be saved via a separate button and automatically initiated in the same way as the user profile references.

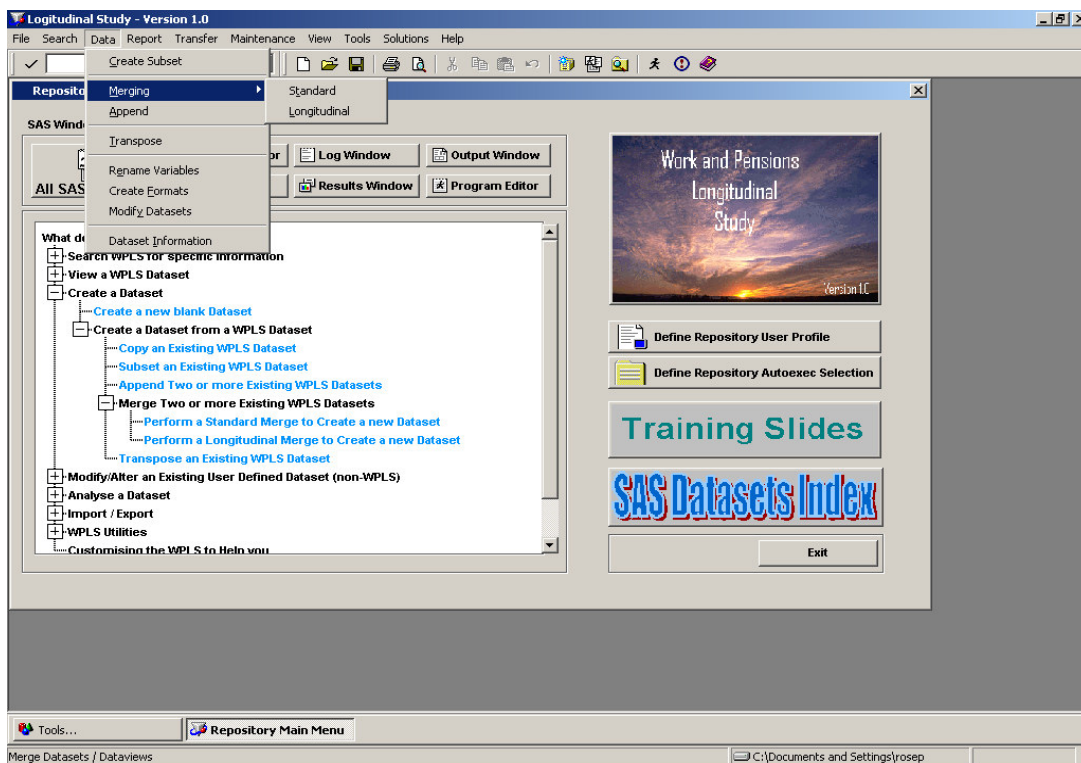
Selection Index



The selection index is a hierarchical drill down list similar to the folder list in MS Explorer. All the various functions can be selected from the selection index by simply clicking on the relevant function. Many are accessible by other routes such as the opening of the standard SAS windows or creation of user profiles. Also pull down menus can be used to access various functions. These therefore provide alternative means of using the interface to make it as user friendly as possible.



Similarly the functions and also the template code library can be directly accessed via pop-up selection lists at the base of the interface.



Training Slides and Dataset Index

The Training Slides button links to a set of documentation used for training users in the use of the interface. This also acts as an online help facility.

The Dataset Index button provides a link to a site on the intranet that lists the metadata behind each benefit \ project type. This avoids time delays from having to access 170 million record datasets on remote servers merely to find a variable name for example.

Summary

In general the interface provides a standardised, streamlined means of analysing the WPLS data by analysts of wide ranging SAS expertise. It is a very popular tool widely used across the analytical community in DWP.

1. Has a wide ranging functionality.
2. Produces and uses efficient, standardised SAS syntax providing high quality, reliable, reproducible results. Ensure results can be confidently compared to similar results achieved across successive time points.
3. Allows novices to SAS to perform complex analyses relatively quickly with minimal debugging
4. Maintains the productivity of the organisation under circumstances that would normally result in a significant drop.
5. Guides the user in the relationships between the various WPLS datasets, the information they hold and how they should be used.
6. Functions can be accessed in a number of different ways allowing the user to choose their favourite method.
7. Can be used to varying degrees from its full extent as an analysis tool to merely an online aid to SAS syntax.
8. Promotes user self improvement in SAS programming

Future Development

1. The users are grouped into teams according to the data they analyse. Therefore each group usually only requires certain combinations or subsets of datasets. At the moment the user has to select the datasets, specify how they should be joined or subset and subsequently process the data prior to any analysis. These specifications can be automated leaving the user to merely select the type of data

they require. The rest will be automatic presenting the user with the relevant data in the best format for their analysis.

2. Also much of the analysis and report production could be automated in the same way. Therefore it may be possible to build a system that produces at least the routine repetitive type of reports at a click of a button and maybe some of the more complicated.
3. It has already been mentioned that as a security feature the SAS log is captured and searched for key words that may suggest misuse of the WPLS data. A separate interface can be produced to automate the key word search. This could be run over night on all the logs produced that day and reports created detailing the findings for further investigation by the security personnel.
4. The original interface on which the WPLS interface was based is currently being converted to be a web based application.

Contact Details

Linking Big (ish) Government Datasets Using SAS and A Few Other Friends

Carol Calvert

Database Services Unit 5 - Destinations
IAD: Information Centre, Department for Work and Pensions,
Kings Court,
80 Hanover St.
Sheffield, S3 7UF
Telephone: 0114 2098 124
Email: Carol.Calvert1@dwp.gsi.gov.uk

Work and Pensions Longitudinal Study (WPLS) Interface

Philip Rose

IAD: Information Centre, Department for Work and Pensions,
Kings Court,
80 Hanover St.
Sheffield, S3 7UF
Telephone: 0114 2098 376
Email: Philip.Rose@dwp.gsi.gov.uk

Annex A – Coverage of Benefits and Programmes

COVERAGE OF BENEFITS AND PROGRAMMES

Current Programmes and Benefits

Future

Basic Skills	ND Young People	BB	
Joint Claims	ND 25+	IB	Surveys
Work Based Learning	ND Lone Parents	ICA	Tax Credits
for Adults	ND Partners	IS	
Employment Zones	ND for Musicians	JSA	
ONE	ND Innovation	SDA	
Action Teams	Fund	WB	
Outreach	ND Disabled		
MEO	Inland Revenue		

Annex B - Soundex

SOUNDEX: The soundex algorithm was that supplied within SAS where SAS help indicates :

"The **SOUNDEX** function encodes a character string according to an algorithm originally developed by Margaret K. Odell and Robert C. Russel (US Patents 1261167 (1918) and 1435663 (1922)). The algorithm is described in Knuth, **The Art of Computer Programming, Volume 3** (See [References](#)).

The **SOUNDEX** function returns a copy of the *argument* encoded by using the following steps.

1. Retain the first letter in the *argument* and discard the following letters:

A E H I O U W Y

2. Assign the following numbers to these classes of letters:

1: B F P V

2: C G J K Q S X Z

3: D T

4: L

5: M N

6: R

3. If two or more adjacent letters have the same classification from Step 2, discard all but the first. (Adjacent refers to the position in the word prior to discarding letters.)

The algorithm described in Knuth adds trailing zeroes and truncates the result to the length of 4. You can perform these operations with other SAS functions.

Annex C – SAS Code for Fuzzy Matching Process

```
*****
* destin.mastmain is the NINo/ORSids that
* are the simpler processing
*
* then do the fuzzy match for those not found on NINO
* mosr efficient to do it here a only need to put on the orcid
* and hence no need to bring full master index record
*****;
%macro match ( found,notfound,infile);
  data &found &notfound;
    length filt5 $40.;
    merge destin.mastmain (in=wanted ) &infile (in=found);
    by nino_body;
  if wanted and found then
    do;
      flag=0;
      if (sex=1 and sexir="M") then flag=sum(flag,1);
      if (sex=2 and sexir="F") then flag=sum(flag,1);
      if
        (scan(postcode,1)||substr(scan(postcode,2),1,1)=scan(pcodeir,1)||substr(sca
n(pcodeir,2),1,1)) then flag=sum(flag,1);
      if (compress(soundex(surname))=compress(soundex(misname))) then
flag=sum(flag,1);
      if ( substr(forename,1,1)= substr(mifname,1,1)) then flag=sum(flag,1);
      if (dob=date_of_birth) then flag=sum(flag,1);
      if flag in ( 4,5) then match="Green";
      if flag =3 then match="Amber";
      if flag =2 then match="Red";
      if flag in (0,1) then match="Nomtch";
      output &found;
    end;
  if (not wanted) and found then
    do;

      filt5=compress(soundex(surname)||substr(forename,1,1)||sexir||scan(pcodeir,
1)||substr(scan(pcodeir,2),1,1)||date_of_birth);

      filt4a=compress(soundex(surname)||substr(forename,1,1)||sexir||scan(pcodeir
,1)||substr(scan(pcodeir,2),1,1));

      filt4b=compress(soundex(surname)||substr(forename,1,1)||sexir||date_of_birt
h);

      filt4c=compress(soundex(surname)||substr(forename,1,1)||scan(pcodeir,1)||su
bstr(scan(pcodeir,2),1,1)||date_of_birth);

      filt4d=compress(soundex(surname)||sexir||scan(pcodeir,1)||substr(scan(pcode
ir,2),1,1)||date_of_birth);

      filt4e=compress(substr(forename,1,1)||sexir||scan(pcodeir,1)||substr(scan(p
codeir,2),1,1)||date_of_birth);
      output &notfound;
    end;
  run;
%mend match;
run;
*****
* Need to only consider those IR cases where the filter
```

```
* gives a unique person instance id- most cases for
* 5 variables less on four, especially that with no date of birth
* and least on three
*****;
*****
* so as for the master file processing
* first get a list of filter person_instance_id combinations
* then check that ensure only take ones where unique
*
* Note that if not unique on 5 will not be on less
*****;
%macro fuzzy (filter,mifile,irfile,mtchtype,mifound);

proc sort data=&irfile nodupkey out=&irfile;
by &filter person_instance_id;
run;

data forfuzzy;
set &irfile;
by &filter person_instance_id;
if first.&filter=last.&filter then output;
run;

data tryfuzzy nogood;
merge forfuzzy (in=okay) &irfile (in=source);
by &filter person_instance_id;
if (okay and source) then output tryfuzzy;
if ( not okay and source) then output nogood;
run;

data found nogood2;
merge tryfuzzy (in=ir) &mifile (in=mi);
by &filter;
if (mi and ir) then
do;
match=&mtchtype;
output found;
end;
if (ir and not mi) then
do;
/* drop orcid*/;
output nogood2;
end;

data found;
set found (where=(&filter ne "."));
run;
*****
* Now tidy up by combining the nogood files
* and the ones we have got a fuzzy match on
* with the original NINo found and not found files
*****;
data &irfile;
set nogood nogood2;
run;

data &mifound;
set &mifound found;
run;

%mend fuzzy;
```

run;

```
%match(destin.xxxid,destin.xxxnid,destin.throwa); run;  
%fuzzy(filt5,destin.mastmain_f5,destin.xxxnid,"GreAmb",destin.xxxid); run;  
%fuzzy(filt4a,destin.mastmain_f4a,destin.xxxnid,"RedAmb",destin.xxxid);  
%fuzzy(filt4b,destin.mastmain_f4b,destin.xxxnid,"AmbRed",destin.xxxid);  
%fuzzy(filt4c,destin.mastmain_f4c,destin.xxxnid,"AmbRed",destin.xxxid);  
%fuzzy(filt4d,destin.mastmain_f4d,destin.xxxnid,"AmbRed",destin.xxxid);  
%fuzzy(filt4e,destin.mastmain_f4e,destin.xxxnid,"AmbRed",destin.xxxid);  
run;
```