# Processing Everything - lessons from comprehensive automated processing of the UK Large Scale Government Surveys

CCSR Working Paper 2008-12

Sam Smith
S.Smith@manchester.ac.uk

A common problem when searching repositories for secondary microdata is finding useful data to meet specific requirements. Variables are a fundamental building block of data analysis and usage. This paper covers the technical implementation and design of a infrastructure underlying a information and cross-references system for finding variables in each file of the 650 (and growing) large-scale UK Government datasets supported by ESDS Government and the Samples of Anonymised Records.

# Processing everything - lessons from comprehensive automated processing of the UK large scale government surveys

Sam Smith
www.ccsr.ac.uk

CCSR Working Paper 2008-12

June 2008

**Abstract**

A common problem when searching repositories for secondary microdata is finding useful data to meet specific requirements. Variables are a fundamental building block of data analysis and usage. This paper covers the technical implementation and design of a infrastructure underlying a information and cross-references system for finding variables[1] in each file of the 650 (and growing) large-scale UK Government datasets supported by ESDS Government[2] and the Samples of Anonymised Records[3].

This paper discusses how decisions made stage have significant benefits, and minor downsides, further along in processing. The paper does not cover the collection or dissemination of data, but talks of the process for enhancing the value of the data and metadata that is already published as part of other processes.

[1]www.ccsr.ac.uk/esds/variables
[2]www.esds.ac.uk/government
[3]www.ccsr.ac.uk/sars/

## Contents

## 1 Introduction

A common problem when secondary analysts search repositories for microdata is finding useful variables to meet specific requirements - variables being the fundamental building block of data analysis and usage. This paper describes the design and technical infrastructures of an extensible platform for data and variable discovery[4] within the 650+ large-scale UK Government datasets, 25 surveys[5] over 30 years that are supported by ESDS[6] Government[7].

The extensive time series of the Government Surveys are one of their advantages, with similar datasets existing on a sub-annual basis since the 1970s for some of the surveys. However while the "Labour Force Survey" has not changed its' name since the 1970s, changes in the survey environment over the last 30 years have been reflected in the surveys themselves, the questions asked, the methods used to ask them etc, not to mention the file formats in which the data are supplied, which have been sequentially converted to the current standard (SPSS) over time.

Social Science data has significant and fundamental differences to large scale datasets produced in other scientific disciplines. While some of our data runs over 30 years, and may be trying to get the same information, over time, both the question and the answer may change. "What is your main job?" is not only going to have many more and different answers now than 20 or 15 years ago, the definition of "main job" has changed over time. These changes can not be represented within the dataset, but are part of the constantly evolving context of the data we're handling. While in a physics dataset the dataset has a relatively simple structure - being a set of (possibly highly complex) readings from experiments or simulations, it is at least predictable. The surveys with which we deal are primarily collected by asking people questions, and the answers returned can be somewhat idiosyncratic based on the differences between people being interviewed.

[4]`www.ccsr.ac.uk/esds/variables/`

[5]`www.esds.ac.uk/government/variables`

[6]the Economic and Social Data Service

[7]`www.esds.ac.uk/government`

## 1.1 Acknowledgements

We must thank all those at ICPSR[8] for inspiring and detailed conversations in 2006. Special thanks must go to Peggy Overcashier and Cole Whitman, who provided clear insight into what was possible given what they were doing, which was far in advance of what we had considered previously. We should also thank Kevin Schurer of the UK Data Archive, and especially Tanvi Desai of the LSE. As ever, Angela Dale, Gillian Meadows, Anthony Rafferty, Jo Wathan and Vanessa Higgins of ESDS Government have provided expert and vital feedback and ideas. ESDS Government must also thank the SARs[9] project, as part of the UK ESRC Census Programme[10] for the infrastructure, designs and time.

## 1.2 Terminology

ESDS Government supports 25 surveys, each of which is a time series of cross-sectional surveys dating back as far as the 1970s. A dataset is one cross-section of one survey, and there can be any number of datasets per year, depending on the survey frequency. These surveys change significantly over time potentially without any name change in variable names. ONS policy (oft imitated) means that any change in a variable generally leads to its name being changed, at least for the more recent surveys, which gives us some level of confidence that two variables in different cross-sections of the same survey are likely to be somewhat comparable. That does not necessarily hold across different datasets. The SARs are similarly designed datasets consisting of anonymised microdata from the 1991 and 2001 censuses.

A univariate distribution gives the range and spread of the values for a variable e.g. by listing frequencies of each value.

[8] `www.icpsr.umich.edu`
[9] `www.ccsr.ac.uk/sars`
[10] `www.census.ac.uk`

# 2 Data Processing

We'll follow the process of the data pipeline[11] through. Although it is not, and can not be, a strictly linear process, and the implementation is fundamentally reentrant and potentially[12] recursive. Appendix A contains an overview of how a single dataset flows through. For example, it is perfectly normal for additional metadata to appear at a later date, after the original data release - the clearest example of this is Nesstar, where ingest into Nesstar can not begin until the data is ready for release, and that release should not be held up because of Nesstar. Reasons such as this are the primary drivers behind the replicability of the processing. We periodically rescan Nesstar and reparse the metadata that it holds and reimport the relevant datasets to pick up any changes to the metadata. Similarly, a dataset may get reissued for one of many reasons, and so we need to re-fetch and update the relevant database entries and regenerate the pages[13].

For reference, a copy of the database structures are included as Appendix B. While they may not be directly clonable for resuse, the relatively simple structure should hopefully give enough information.

## 2.1 Data download

Data download takes place via a script which follows the standard data registration and download procedures followed by all UK academic users in their web browser. We deliberately behave as a "normal user" and additionally exclude datasets with any extra restrictions on them over and above the standard license agreement. Inclusion of these "Special License" datasets is technically possible, although extra discussions with data providers would need to take place.

## 2.2 Pre-Processing

While recent data is generally of a high standard, our portfolio includes data that was created over 30 years ago, and has been run through a variety of conversion processes over the course of that time, as software has changed and new formats were required. While data conversion has been of a high standard and opened correctly in all newer software, with the advent of a wider range of software (such as R), it is no longer enough

[11]See Cole Whitman's IASSIST publication and related reading at `http://icpsr.umich.edu/ICPSR/org/publications/staff/ProcessMapping.pdf`

[12]but carefully

[13]We'll consider the effects of why a dataset may be reissued in a few pages

for the data file to open, the data file has to match the format specification. SPSS has the advantage of being a ubiquitous and well documented format.

A simple example of conversion problems is taking an SPSS export file (`.exp` extension) and renaming it to have a `.por` extension, and with no other changes, SPSS will open it correctly. On saving the document, SPSS will create a valid portable file, but should be closed without saving in SPSS, as the unsaved file will not follow the specification for SPSS portable files (as all you did was rename it) so other software may not open it. Previously, this has not been a significant problem, however, with the proliferation of different software options for analysis (the recent rise of R and M-Plus for example), there should be no expectation that the software that users are opening data in is the same software it was originally created for. This is going to be a major issue as e-Social Science becomes more mainstream - SPSS is not the dominant platform on the grid, and open-source based solutions are used for opening datafiles.

While the process downloads and use SPSS data as the input, it does not generally use the SPSS software itself as part of the normal workflow. For older datasets, a script is run to open adn save all datasets to ensure the formats are as described. While there were several thousand files to be processed, this was easily accomplished in a single run overnight. During the process, we found that some datasets had failed to open, and the previous dataset had been saved instead. While this only happened a couple of times, and was completely unrepeatable, it was a problem. Therefore an additional check was preformed by creating an MD5 checksum of each file processed and looking for duplicates to ensure that any files so affected could be resaved through another syntax file. The reasons for this problem remain unexplained.

The main output of this stage is the dataset expanded from the zip archive, having been resaved. Due to the above process, it is not enough to only permanently store the original zip file, but the processed output should be kept to enable reprocessing. The other output created at this stage is a list of the datafiles as unzipped. This has the advantage of allowing any data conversions to be transparent to higher layers of processing - they simply ignore files that aren't listed. As we were running all the historic data through SPSS anyway to clean up the file formats, we took the opportunity to resave the file as a SPSS .sav format, rather than portable. As all new data comes as SPSS sav files, it ensured all input is in one standardised format and reduced the number of paths through processing that data follows.

## 2.3 Univariate Data Processing

The core of any variable information system is the production of a univariate list of values.

Using one of two readily available scripts, we produce a list of variable names, values and labels within each file. While these scripts do not provide any distribution information, they provide a the core list of variables and files.

To obtain the values and labels for the data from the SPSS files, we use scripts to provide the variable level information about labelling. SPSS portable files are either converted to SPSS .sav format[14], or run through the *xlabels* script written by Frank Stetzer of UW Milwaukee[15]. SPSS .sav files are processed by the *spssread.pl* script by Scott Czepiel[16] which produces a tab seperated file of information about the variables or values. These two outputs are merged to provide the univariate distribution information, along with variable and value labels. That is the extent of the metadata that is stored in most SPSS files.

This information is then used to generated an R command file which opens the dataset (via the foreign library), and runs the summary command on each variable e.g.

```
print (summary(TPBEN38), print.gap=250);
```

This produces a univariate distribution for each variable in the dataset, with the print.gap argument ensuring that there is one value per line, which aids in parsing the output. The summary function helpfully will produce useful output for variables it believes are continuous, as well as a standard univariate information for discrete variables. That distinction can be detected and saved as useful metadata for display later in the processing.

The univariate information, including the filename/variablename, label, and value information is then stored on disc in a simple text file which is substantially similar to the format used by Survey Documentation and Analysis System (SDA) at Berkeley[17] While DDI is an excellent interchange format for sharing data between organisations and for forward archiving of final metadata, it is both hierarchical and complicated in many ways that the univariate distributions aren't. As a result, while we can both import from and export to DDI without a problem, internally, we use something that's only as complicated as we need.

The metadata file is always stored with a set of defined filenames (e.g. values) which allows other processes to use the file information. This information could as easily be stored in a database with suitable priorities allowing some items to override others.

[14]for why this happens see the detailed process

[15]and available from `ftp.uwm.edu/pub/stetzer`

[16]`http://czep.net/data/spssread/`

[17]`http://sda.berkeley.edu` - a great system that we don't use

## 2.4 Additional Metadata sourcing

We are fortunate, to get a large am ount of documentation from our data providers, covering broad aspects of the data, its' collection, derivation and context. While this is a good thing, it can be a relatively large haystack within which to start searching for the needle needed to start sewing the data together for specific research purposes.

As all our datasets have defined catalogue names, and all the documentation is accessible via the UKDA website which we can index, and then download all the PDFs for that dataset. Once we have them downloaded, we can easily convert them to an xml format, and then search each xml file for each of the variables names we have already found. This allows us to produce a variable based index to all the documentation, giving the document file names and page numbers where that variable is referenced. Keeping a simple count of the references found gives us one way to measure the "importance" of any page in the documentation, as a page which mentions the variable name more is likely to be more relevant. Any page which references the variable at all may be interesting (as the researcher may find a derived variable more useful than the main one they originally looked for).

However, no matter how much good metadata exists locked away in databases, it is only useful and usable if it is able to be accessed in an automated way.

## 2.5 Nesstar, DDI, and other external services

Both the ESDS and the SARs projects are fortunate in having a range of complimentary services surrounding them, which provide different information and services to users, and to which we send users for pieces of information - such as to the Question Bank for the survey questionnaires. The use of common and consistent acronyms and hence predictable URLs means that it easy for us to link information on any dataset held by any of these compliemntary services, without forcing users to search or a high degree of integration effort.

The UKDA, who disseminate the data we support, do significant amounts of work on some datasets to add metadata and put them into the Nesstar system[18]. This work includes question texts and additional metadata such as universe information based on routing. All published to Nesstar, which makes available an XML file of datasets it contains and the associated URL to download the DDI file for each of them. This allows an automatic process to run to download the latest DDI metadata and merge it with

[18]`http://nesstar.esds.ac.uk/`

the information we have created. This also allows us to cross check the DDI data with the variable information output from our scripts, and then investigate any differences. While the DDI contains univariate distribution information, it is not available for all of our datasets, so for simplicity and consistency of processing, it makes sense to use a single system available for all our datasets. Additionally, due to the extra work required to put a dataset into Nesstar, this only happens a short time after the dataset becomes available for download, so when a dataset is initially processed, there may be no Nesstar version available.

Creating the system so that it can re-enter the build process and reconstruct any part of itself allows for asynchronous updates from external services and makes an entire rebuild be easily run. There are then two scripts which load all metadata for a dataset into the database, and then commence a rebuild of the relevant webpages and their dependencies.

## 3 Platform

Once the webpages have been built and are loaded onto the webserver, standardised, predictable URLs based on the survey, dataset and variable (with full information and cross-linking where appropriate[19]).

This allows Internet search engines to index all our content, and make all of the combined metadata available without changing their normal working practices - which ends to be searching via Google. The bulk of search traffic appear only once or twice, with people searching for variable names[20]

We can also look across surveys and datasets to find variables with similar names elsewhere, which may be candidates for either usage or merging, based on reversing the process by which variable names were changed, and matching the resulting stem. For example, the variable *actwkdy*2 is highly likely to be related to both the variable *actwkdy*1 in the same dataset, and the variable *actwkdy*2 in a different dataset. Precomputing these matches can take a few minutes per dataset, even after the obvious optimisations

[19] `http://www.ccsr.ac.uk/esds/variables/lfs/lfs5441/actwkdy2/`

[20] e.g. actwkdy2

## 3.1 Building on the platform

As we have a set of pages for each variable, we can then use those pages as a basis for more advanced services. When you can link to a variable to find out all the information about it, you can offer a search function which lets users select the diverse topics which they want a dataset to cover, and then providing them with a list of datasets where the data itself matches those terms.

While it took a few iterations to develop a method of displaying such data in a useful and accessible way, once the script was written, it was a simple matter of rerunning the build script for each survey and then this service was available for all the datasets we support[21]. At negligible extra work, we were able to save our helpdesk staff a significant quantity of time and complex work for some of our more involved queries on data matching. More importantly, it also put that tool in the hands of all researchers, allowing them to answer those questions themselves, and not ask our helpdesk the question at all, allowing staff time to be used on more complex queries.

For example, reusing the simple information of which variables appear in multiple files, allows us to produce a table of all variables which are in multiple files, and which files they're in, which gives researchers an invaluable and unique tool for easily visualising the options for matching across datasets [22] in a way which is otherwise impossible. While users may know the major datasets which cover their areas of interest, there may be additional datasets which match those terms for other reasons, but those topics were not their primary area of interest, and those terms do not appear in the hand created metadata.

Within a single dataset there may be many data files (the most we have is over 60) and need to be joined before the files can be used together. Variables having the same name in multiple files implies that they perform some sort of linking function, and putting this information in a publicly visible table is useful both for quickly helping researchers doing data linkages, and data support staff when answering questions.

Developing new services is a relatively simple process as the underlying pages to link to - containing the data of interest to users, is already in place. Additional functionality is generally developing new and innovative ways of directing users to that those pages via a range of different services offering diverse routes.

[21]obviously, we skipped the datasets which consist of only one file

[22]`http://www.ccsr.ac.uk/esds/surveyfinder/`

## 3.2 The Internet Environment

It is increasingly the case that users are getting both happier with online delivery of resources, and also requiring increased usage of those resources, in a way which researchers expect. Since exposing the pages to google, prior to launch, we were seeing significant numbers of users finding the information they wanted due to their presence in search engines - people were googling the variable names that they were looking for.

While this seems completely natural behaviour to those who are comfortable using internet services. It was something of a suprise when we ran variable name based search in google, that until our pages existed, no relevant information was found from either ourselves or other relevant organisations. We also had no idea that they were even looking for it as no relevant services came up. While some of this information was available online if you knew where to look, it was not findable by google, nor any of google's users. Walled gardens can be nice to look at, and networks of walled gardens can be nicer, but people need to know they're there and how to get through the door when they do.

Once we have everything in a database, and we add relevant date information to datasets when they appear, we can offer email and RSS alert services for new datasets based on their full contents, rather than just on the survey that the dataset contains. This enables us to tell users about a dataset is available that contains "Pakistani", "part-time" combined with any other words that meet their research interests, rather than just telling users there's a new Labour Force Survey out and leaving users to find out whether it's useful for their questions.

When that information on pages viewed, and the web server referrer logs which include the search queries that users entered into google or elsewhere which then lead them onto our pages. Looking at this information on an aggregate basis in terms of overall search traffic, we currently see a wide range of search terms, but rarely - generally following a long tail distribution.

## 3.3 The benefits and drawbacks of this approach

The benefit of a hierarchy of scripts which run a number of specific components of the process allows for individual process stages to rebuild as much or as little data as they need. The single main script (called *go*) runs on a regular basis downloading all new datasets and running them through the full pipeline, calling relevant other scripts in the right order for the right datasets. Individual subsystems (such as nesstar) can either run asynchronously or from those same scripts. There's a single script to load all metadata into the database, and another (with a number of slave-scripts) to rebuild the webpages for a dataset. This made possible an iterative approach of getting something working followed by continuous and ongoing improvements to the services we offer.

This makes it relatively easy to do as much or as little work on a dataset as required. It is also exceptionally easy to reprocess one or all datasets without any extra work.

While it is possible to generate each page dynamically on a per request basis, reading information and will regenerate pages as needed. Much of our data is subject to re-issue due to errors, omissions, or a variety of reasons such as population rebasing. This reissue can be full or partial, and may or may not be based on the same original dataset, or a manual reextraction from some other database. As a result, it is not unknown for there to be oversights in an updated dataset, where a variable was added on at a late point in a previous issue, and then forgotten in the latest issue. As the database represents the state of the dataset now (rather than including all historical information for variables dropped), pages for variables that are no longer available will disappear. This may be the desired behaviour where data has been deliberately removed, for accidents and oversights it is not.

We therefore generate over 750,000 static pages which then get overwritten when data is updated (and when a variable is dropped, the page still exists, but is not listed in any indices or searchable). While this can be a somewhat time consuming process to update, it does mean that we have some infrastructure in place which makes it easy to find variables that have been accidentally dropped. Given the long-term nature of our project, and the expectation that the data we support, collected 30 years ago, will still have research value in another 30 years, this permanence is a feature, rather than than a bug. This also has the fundamental and significant benefit of decoupling the system outputs from the language in which they're implemented. While all our code is written in Perl and shell scripts, that could all be replaced, transparently, without any visibility to users should there be a reason to.

As all of these systems are made available via our website to the whole community, not just our support staff, this has the benefit that when such a query comes in, we can easily both send the user the link with the answer in it, and also encourage them to use that service themselves in future so that they don't need to wait for our helpdesk to reply. This frees up staff time to work on added value rather than repetitive questions. Similarly, the platform allows for the creation of small custom services to regular classes of queries as it is exceptionally easy to provide a link onwards to the relevant variable pages.

## 4 Summary

One of the main aims of this was to make metadata and information from one of our surveys taken 20 years ago as accessible as our latest release. While this will never be possible, as we simply have less metadata, it succeeds in making as much available as we do have; starting from the data files themselves, and building everything on the fundamental building block of quantitative social science research - the data.

The web pages produces are static and permanent, ensuring that even through data reissue and any potential system recreation, any historic data must be explicitly deleted, rather than explicitly maintained - and the system will not stop working through neglect, but will simply stop updating. This also aids with the scalability issues, in that older data does not take up a significant resource (it's just bits on disc)

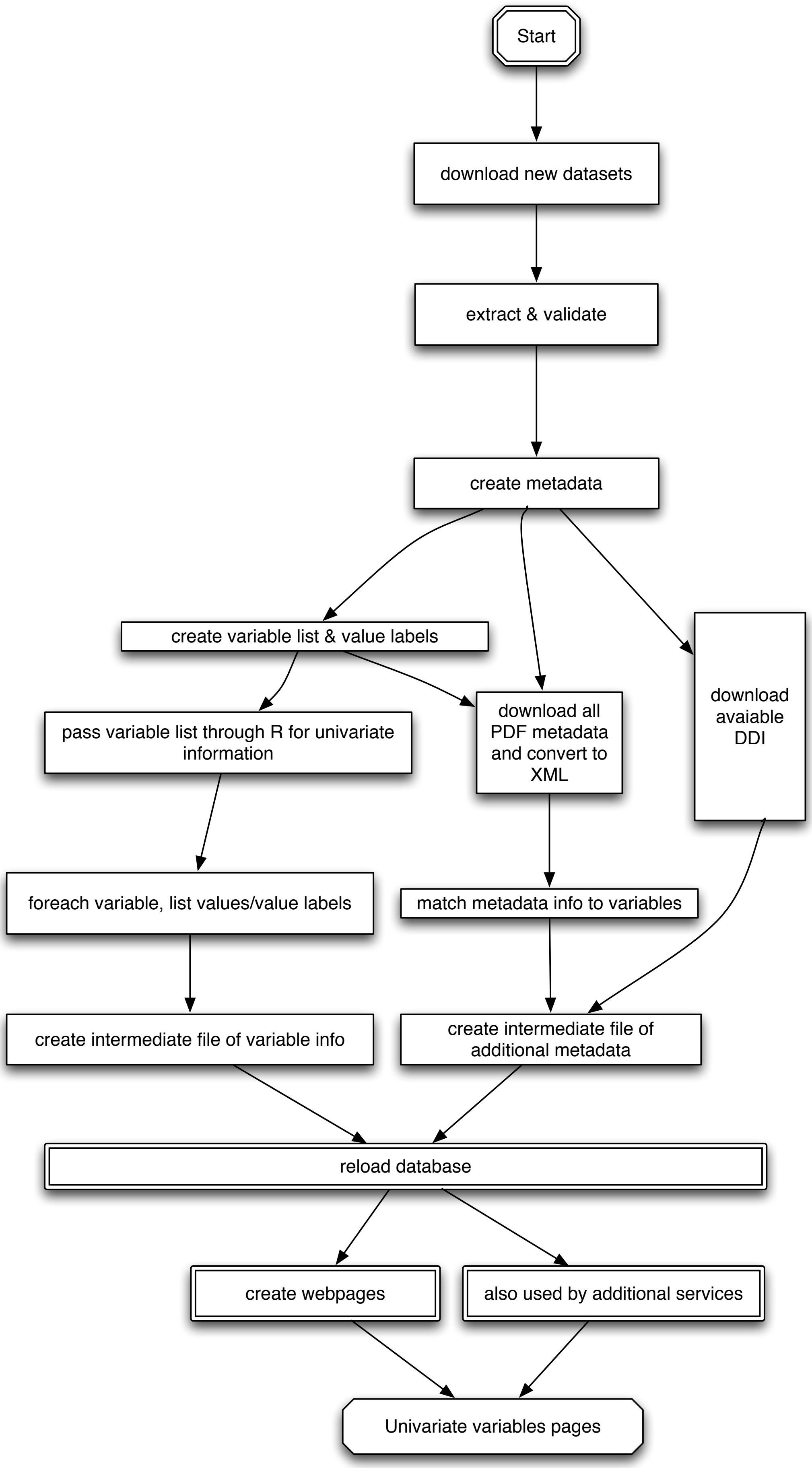
The iterative design process of continuous improvement was made possible via the data pipeline model. Sections could be rerun, or the entire data holding completely reprocessed, as improvements were made, at negligible extra effort.

As for future developments, the system is expected to continue to scale as we add more data into it. While there are likely to be limits of the systems on which this runs, the system is expected to continue to scale until we run out of data to put into it.

We leave all metadata easily accessible and open for Internet search engines to index all our content. Users increasingly require online resources in a way which they expect where they are already. Since exposing the pages to google, prior to launch, we were seeing significant numbers of users finding the information they wanted due to their presence in search engines - people were googling the variable names that they were looking for. Previously, we had no way of even knowing that they were doing this, let alone not getting back useful information.

Most importantly, we aim to solve users' queries. For some questions that previously have required a conversation with the helpdesk, we now provide a public and reusable solution for users to solve their own queries, which also helps our staff when we do get queries.

## Appendix A: Workflow Diagram


Start
download new datasets
extract & validate
create metadata
create variable list & value labels
download all PDF metadata and convert to XML
download avaiable DDI
pass variable list through R for univariate information
foreach variable, list values/value labels
match metadata info to variables
create intermediate file of variable info
create intermediate file of additional metadata
reload database
create webpages
also used by additional services
Univariate variables pages

## 6 Appendix B: Data Structures

The database table structures behind the system are below.

```
CREATE TABLE ‘files‘ (
‘surveyid‘ int(10) unsigned NOT NULL auto_increment,
‘filename‘ varchar(200) NOT NULL,
‘ukda_surveyno‘ int(10) unsigned NOT NULL default ’0’,
‘survey_name‘ varchar(200) NOT NULL default ’’,
‘url‘ varchar(200) default NULL,
‘survey_acronym‘ varchar(10) NOT NULL default ’’,
‘ddi_survey_id‘ varchar(30) NOT NULL default ’’,
‘project‘ varchar(14) NOT NULL default ’missing’,
‘year‘ smallint(5) unsigned default NULL,
PRIMARY KEY (‘surveyid‘),
KEY ‘survey_name‘ (‘survey_name‘,‘filename‘,‘surveyid‘)
);

-- the only table that is populated from outside the system.
-- Various parts of this table are guessed from others, but
-- open to be corrected by administrators.

-- surveyid is the internal id number for a dataset
-- ukda_surveyno is the catalogue number at the UKDA
-- survey_name is the long title of the dataset
-- url is the generated URL where pages will live
-- survey_acronym is the acronym for the series (e.g.  lfs)
-- ddi_survey_id is the reference used in DDI we import
-- project is set to ’sars’ or ’esds’
-- year is the year of data processing - for date based search


CREATE TABLE ‘file_info‘ (
‘fileinfoid‘ int(10) unsigned NOT NULL auto_increment,
‘surveyid‘ int(10) unsigned NOT NULL,
‘info_key‘ varchar(30) NOT NULL,
‘info_value‘ text,
PRIMARY KEY (‘fileinfoid‘),
KEY ‘finfo‘ (‘surveyid‘,‘info_key‘)
);

-- store for key/value pairs that we can pull from the metadata automatically
```

```
--
-- Table structure for table `variables`
--

CREATE TABLE `variables` (
`variableid` int(10) unsigned NOT NULL auto_increment,
`variable_name` varchar(30) NOT NULL,
`variable_label` varchar(100) NOT NULL,
`surveyid` int(10) unsigned NOT NULL,
`url` varchar(100) default '',
`filename` varchar(50) NOT NULL default '',
`stillthere` tinyint(3) unsigned default '0',
PRIMARY KEY (`variableid`),
KEY `main` (`variable_name`,`stillthere`,`surveyid`,`variableid`),
KEY `search` (`surveyid`,`variable_name`,`stillthere`,`variableid`),
FULLTEXT KEY `ft2` (`variable_name`,`variable_label`)
);


--
-- Table structure for table `var_values`
--

CREATE TABLE `var_values` (
`valueid` int(10) unsigned NOT NULL auto_increment,
`value_name` varchar(100) NOT NULL,
`value` varchar(20) default NULL,
`percent` varchar(10) default NULL,
`hide_count` tinyint(3) unsigned NOT NULL default '0',
`count` int(10) unsigned NOT NULL default '0',
`variableid` int(10) unsigned NOT NULL,
`surveyid` int(10) unsigned NOT NULL,
PRIMARY KEY (`valueid`),
KEY `mainlookup_varvalues` (`value_name`,`valueid`),
KEY `main` (`surveyid`,`variableid`,`valueid`),
KEY `variableid` (`variableid`,`surveyid`,`valueid`),
KEY `value_name` (`value_name`,`variableid`,`surveyid`),
KEY `s3` (`valueid`,`surveyid`,`variableid`),
FULLTEXT KEY `ft2` (`value_name`,`value`)
);

-- the table that contains all the information about values in datasets
```

```
--
-- Table structure for table `var_lookup`
--

CREATE TABLE `var_lookup` (
`varlookupid` int(10) unsigned NOT NULL auto_increment,
`survey_name` varchar(10) NOT NULL,
`variable_name` varchar(30) NOT NULL,
PRIMARY KEY (`varlookupid`),
KEY `main` (`survey_name`,`variable_name`,`varlookupid`)
);


--
-- Table structure for table `matches`
--

CREATE TABLE `matches` (
`matchid` int(10) unsigned NOT NULL auto_increment,
`surveyid1` int(10) unsigned NOT NULL,
`surveyid2` int(10) unsigned NOT NULL,
`varname1` char(30) NOT NULL default '',
`varname2` char(30) NOT NULL default '',
`manual` tinyint(3) unsigned NOT NULL default '0',
`negative_match` tinyint(3) unsigned NOT NULL default '0',
PRIMARY KEY (`matchid`),
KEY `mainmatch` (`surveyid1`,`surveyid2`,`varname1`,`varname2`,`negative_match`),
KEY `s1` (`surveyid1`,`surveyid2`,`negative_match`)
);

-- manual matches are those added by hand.  negative_match allows items which
-- do not actually match, but appear that they do, to be flagged and excluded
-- by only selecting matches where that flag is not set in output, and ignoring
-- the setting at input
```

```
--
-- Table structure for table ‘metadata‘
--

CREATE TABLE ‘metadata‘ (
‘metadataid‘ int(10) unsigned NOT NULL auto_increment,
‘valueid‘ int(10) unsigned NOT NULL,
‘variableid‘ int(10) unsigned NOT NULL,
‘metadata_key‘ varchar(200) default NULL,
‘metadata_value‘ text,
‘url‘ varchar(255) default NULL,
‘surveyid‘ int(10) unsigned NOT NULL,
PRIMARY KEY (‘metadataid‘),
KEY ‘main‘ (‘surveyid‘,‘variableid‘,‘metadata_key‘),
KEY ‘var‘ (‘variableid‘,‘valueid‘,‘surveyid‘,‘metadataid‘),
KEY ‘missing‘ (‘metadata_key‘,‘variableid‘),
FULLTEXT KEY ‘ft1‘ (‘metadata_value‘)
);

--- table for storing metadata (key/value pairs) and associated information
```