Understanding current practice and systems for analysis of integrated data in the Stats NZ Data Lab

Peter Ellis - Principal Data Scientist, Stats NZ

21 September 2017

Version	Date	Comment
0.1	22/8/2017	First discussion with advisory group
0.2	3/9/2017	First complete draft
0.3	4/9/2017	Incorporate advisory group discussion
0.4	11/9/2017	Incorporate feedback from reviewers
0.5	12/9/2017	Minor additions
1.0	21/9/2017	Incorporate feedback from Stats NZ management

1 Executive Summary

1.1 An attempt to understand current practice

The purpose of this report is to inform options and design thinking for the “IDI 2” project, and in particular help scope the possible needs and opportunities for information and analytical “layers” for the IDI. I outline researchers’ current approaches to analysis of the data in the Integrated Data Infrastructure (IDI) and the Longitudinal Business Database (LBD). In the process of understanding how researchers use the integrated data, I have made observations of the databases and analytical environment that holds that data. The “summary information layers and confidentiality solution” workstream is one of four workstreams under the IDI 2 project, the others being “access pathways”, “fundamental redesign”, and “scalable cloud-based infrastructure”

This report represents the views of the author and should not be attributed to Stats NZ.

I set out to answer five questions, which are used to structure this report, attempting to understand:

- the data itself, the infrastructure that holds it, and the analytical environment available to researchers
- what researchers are trying to do
- how they use the toolkit available to them
- how they work as a community, sharing good practice, code, and joint development of assets
- opportunities for improvement, which may or may not be summary information layers.

The method used in producing this report included:

- setting a framework of “what good might look like” for integrated data and its analysis
- a small number of interviews with active IDI and LBD researchers
- observation and analysis of the databases and analytical environment that host the IDI and LBD
- analysis of file metadata (eg file types, numbers and sizes) in 173 researchers’ “project folders”
- analysis of code excerpts published on the Data Lab “Wiki” and elsewhere

- an overview review of the actual code used in a large number of past and present projects, and in-depth investigation (including running code to re-create analyses) of eleven of them.

1.2 Understanding the integrated data and its analytical environment

Section 3 of this report describes the Data Lab environment and the researchers' databases in it.

In data warehousing terms, the IDI and LBD are presentation layer datamarts. The main parts of the IDI and LBD are three distinct SQL Server databases on separate machines, supported by several secondary databases. They are large (380GB and 200GB respectively). But what is more distinctive than their size on disk is their complexity - 553 tables in the two databases that comprise the IDI and 126 in the mainstream part of the LBD.

The IDI and the LBD are hugely important and impressive achievements for New Zealand. Both Stats NZ and the researcher community are to be congratulated for the contribution to understanding of New Zealand social and economic issues.

The integrated data environment Stats NZ hosts for researchers has been greatly improved in recent years. Major improvements have included:

- more usable form for the tables linking codes to their descriptors (called by Stats NZ and researchers "metadata")
- improvement of documentation through the "Wiki" in the Data Lab and the beginning of substantial sharing of code between researchers
- expansion of analytical capacity by an additional SAS server and the first RStudio server
- expansion of derived tables re-presenting data in more usable form such as the "all sources" ethnicity table, the summary address notification table, and the summary income table
- ongoing addition of new data from new sources
- expansion of file server capacity for a larger body of researchers
- successful recovery and transition to a reliable and scalable cloud-based supplier after the November 2016 earthquake

However, compared to widely accepted good practice standards for analytical and data warehousing environments, much remains to be done to help researchers and Stats NZ deliver on their objectives:

- the analytical environment falls short of professional standards in that it lacks version control software - a planned project to implement such software was put on hold, along with other development work, following the 2016 Kaikoura earthquake and subsequent reprioritisation of Stats NZ's work programme
- the databases holding the IDI and LBD are neither designed nor tuned for performance, and could deliver data quicker than they do
- the data models are both ad hoc and incomplete (for example, the full text of many survey questions are not available as tables in the database), making analysis more complex than it needs to be
- the system is not set up to manage slowly changing dimensions such as changing classifications, and this is a major challenge for researchers and obstacle to reproducibility and building a growing corpus of sustainable code
- system performance is inconsistent and at times very slow, or breaking down completely, due to the pressure being put on the network by excessive data movement and reading and writing to the main file server.

Individual snapshots of the IDI and LBD by themselves don't qualify as "big data" in their current state. The traditional relational database management system they are in is well suited to the task of the layer presented to researchers, and SQL Server is a good tool for data of this sort. There could be advantages to big data tools featuring at some point in their production process, particularly in terms of further rationalising of Stats NZ data pipelines, but it isn't necessitated by the size of the final datamart considered by themselves. A relational layer to be interrogated by a SQL-like language is essential for their analysis.

Understanding how researchers are using the IDI and LBD gives us some insight into prioritisation and sequencing for further improvements in this space.

1.3 Most analysis is frequency counts and cross tabs

Section 4 of the report looks at researchers' aims and approach to analysis.

A little less than half of the leads of research projects are from government departments, but government has some of the largest "projects" and probably constitutes more than half of the work being done.

The analysis that is undertaken with the IDI and LBD can be categorised into four types:

a. estimating populations (eg the number of people on benefit living in area 'x'), with simple cross tabs and frequency counts
b. research and evaluations with social science or public policy aims to advance knowledge, using variants of regression models
c. development and deployment of models for purpose of scenario exploration, including by large "micro-simulations" or simpler aggregate models to inform cost benefit analysis
d. meta-research into the opportunities and limits of administrative data for understanding populations, such as the Stats NZ "Census Transformation" project and the MBIE "Housing Affordability Tier 1 Statistic" project

Many "projects" do two or more of these types of analysis, but most have one type that dominates. Most analysis observed in this exercise was of "type a", and simple cross tabs and frequency counts dominate in the output released through the checking process. Type "b" - chunky research and evaluation projects - is the original archetypal IDI research project around which systems such as researcher applications and output checking have evolved, but is now a minority (still substantial) of the analysis actually undertaken in the Data Lab.

The four types of analysis feed off eachother. In particular, simple population estimates ("type a") depend on the success of "type d" projects in delivering convenient and reliable ways of inferring to populations from incomplete administrative data. Models for exploring scenarios ("type c") depend on successfully validated social science knowledge ("type b").

Despite the different aims and levels of complexity of different projects, all research projects looked at for this exercise showed signs of following (and iterating through multiple times) a common sequence:

1. exploration
2. preparation of a dataset through filtering, mutating variables and aggregation
3. analysis, ranging from simple counts through to machine learning
4. dissemination, ranging from random rounding a few numbers in Excel through to development of interactive web apps

Projects vary greatly in size. The largest have tens of thousands of files and hundreds of folders on the file server. Several have more than a terabyte each of data saved on the file server (the IDI in total is about 0.4 terabytes). The smallest active projects have a single researcher in the Data Lab (although often partnered with subject matter experts outside), a dozen files, and a small number of clearly defined outputs.

1.4 SAS dominates usage, but SQL is really the foundation

Section 5 of the report looks at how researchers use the analytical tools available to them - SQL, SAS, R and Stata.

All researchers use multiple tools to obtain and analyse data. The big majority of work and elapsed time is in the first step of data management and preparation. This is to be expected with such large, complex databases. A generic approach that describes IDI and LBD research includes:

1. SQL code for getting the data together and extracting it, during exploration and data preparation
2. Code in one of SAS, Stata or R for further reshaping and then for statistical and econometric analysis
3. Dissemination, usually via Excel for the confidentialisation and output checking processes

This approach is obscured somewhat by researchers' differing preferences for development environment and the blurred line between steps 1 and 2. Most SQL code is probably actually written within SAS Enterprise Guide, and researchers think of themselves as writing SAS programs even though a large portion of the code is SQL wrapped up with `PROC SQL` steps. SAS is the dominant tool, particularly in the IDI. Nearly all the legacy code for the IDI is a SAS / SQL hybrid in the same scripts, whereas for the LBD the SQL for data management is generally more cleanly separated from the use of Stata, SAS or R for analysis.

The particular way researchers use SAS in the given architecture is a major contributor to performance challenges. Large amounts of data move along the narrow "pipe" of the network from the database servers to the SAS servers, are processed there, then sent along the network again to be saved on the file server (or, less frequently, the "sandpit" of the database server). Enormous amounts of data (including 8 terabytes in SAS format) are individual files on the file servers with uncertain lineage, definition or use. The problem might be partly resolved by moving more analysis onto the database server with code written in straight SQL, but individual researchers with a legacy of code in SAS and more familiarity with it than with SQL have little motivation or opportunity to do this. This method of using SAS made more sense several years ago when the file server and SAS server were the same machine, and "metadata" wasn't available in the database and hence was included in the analytical tool by being hard coded into SAS programs.

There is more instability in the database design than is necessary to accommodate real world change. Table and column names and types change; and some of the dimension reference data is in a separate database to the main data, when it exists in a database at all. Changes to dimensions and classifications are handled by changes to the database design (eg adding in new tables with date-stamped names) rather than by extending data within a robust time-invariant data model as is good practice. Adding new data, including of a standard repeat type (eg a new survey) requires design work, not just new values in fact and dimensions. Because of this instability and the lack of code version control, researchers cannot be sure to reproduce any particular piece of data. Because of these environment limitations, analytical programs contain numerous ad hoc, context-specific and time-specific work-arounds and there is no legacy code base for reproducible research.

1.5 A vibrant community that hasn't worked out how to work together yet

Section 6 of the report looks at how the research community has evolved and works together.

Code that has been shared between researchers (mostly on the Discussion Board on the in-Data Lab "Wiki") is good quality and covers an interesting range of data preparation tasks (with many fewer relating to statistical analysis) that could form a good introduction for new users to the IDI and LBD. But even experienced users find the Discussion Board frustrating, difficult to search, and unlikely to help their particular problems; it's usually easier to start from scratch or ask an individual expert if they know who they are.

So big steps forward have been made in sharing code for the IDI (but not as much for the LBD) in the last couple of years, but most researchers are still going it alone with minimal help from others, other than their team if they are lucky enough to have one. Collaboration across projects is extremely difficult because of the Data Lab security provisions. These security provisions exist for good reason, but have side effects such as difficulties in sharing code and discussing techniques and results.

Many multi-researcher projects have their own library of shared functionality - usually SAS macros and library references that are called universally at the beginning of an analytical task. An adaption of an approach to code originally from Treasury is used by several projects, and many snippets of shared code only work if an analyst sets up their environment to mimic a generic Treasury-like folder system and sequence of analysis. This is a good thing that has contributed to a common style and approach for some projects including some discipline in project-wide standard library names and macros, but adds to communication difficulties particularly for new and individual researchers.

Because most code is only shared as detached files, rather than a repository of code, and with limitations in extending SAS in a portable context-independent way, very few pieces of shared code run "out of the box". Sharing code is constrained by the lack of version control software in several ways:

- harder to track releases and versions of dependencies ("if this depends on Treasury's standard macros, which version of them?")
- harder to share a more complex repository of code, because Sharepoint (which the Discussion Board is on) isn't built to do this in the way GitLab or GitHub is
- harder for researchers to iteratively troubleshoot code during a phase of adapting it to their own environment

Even the best managed projects have many variant copies of key scripts, sometimes many hundreds of copies. Most multi-researcher projects have large amounts of project code (ie not just personal exploration, training, and development) in folders named after individuals with no clear path to merging with the master branch of a definitive, corporately owned code base. In the absence of version control this is understandable and perhaps even inevitable. At least four projects have obviously tried to get around this problem by mandating the use of a team rather than individual code base, with mixed levels of frustration and success.

1.6 There are some big opportunities

This paper was written as background for a coming options paper on “layers” for the IDI. There are some definite opportunities in this space. But I also found ample room for improvement in other areas, some more or less related, which I assess as being of at least equal if not higher priority for consideration. More detail is provided in section 7.

The top priority from the perspective of this paper is a professional-quality version control system, to help development in general, in teams in particular, and to make it easier to manage code sharing and joint development. A plan exists for implementation of GitLab, it just needs to be finalised, resourced and implemented. The key issue is the need for this work to be prioritised as it relies on wider Stats NZ resources and sits within wider organisation prioritisation processes.

A second low hanging fruit is some tuning of the existing database to make SQL queries run faster. This would involve a sprint jointly by Digital Business Services and Integrated Data to identify a range of representative queries and make the tweaks to the databases to improve them. For example, some queries that take several minutes to run - a challenge for researchers trying slightly different versions many times - might be able to be improved to running in a few seconds with the right indexes added. Part of the work would be to identify where the biggest pay-offs are for such tuning.

My third identified possibility has more in common with the idea of layers and is the possibility of creating shared *functionality* for the IDI in particular, to execute commonly wanted tasks for different population groups. Such functionality would probably be developed once in SQL and executed on the database server for efficiency, but have light front ends in SAS, R and Stata.

A key element in thinking of any actual persisting layers of data is Stats NZ’s ongoing Census Transformation project, which is systematically tackling a range of questions which could produce useful re-usable datasets in the IDI. This has seen good results already with the “estimated resident population” table and the “ethnicity by ranked source” table. These and related matters (eg weights to match IDI results to official statistics) are important steps towards the long term goal of a relatively simple query of the IDI database returning estimates that are highly granular, good quality, and consistent with published aggregates.

There is a long term possibility of a useful layer for users outside the Data Lab environment, but its feasibility depends on more fundamental design issues. I will explore this issue, and the other layer-related matters, in my next paper on options regarding layers.

My view of the LBD priorities from a user perspective are:

- complete outstanding transformations of the data such as rationalising and combining multiple waves of the Business Operations Survey
- the so-called “metadata” variables such as full text of survey questions taken out of PDF and Excel documents where they reside and placed in actual tables in the database (not just human-readable stand-alone documents on the Wiki as they currently are, although this is a start)
- including data on survey sample design and the populations they are weighted to in the database
- to catch up with the IDI in terms of sharing code and joint development
- a means - again, dependent on version control and code sharing - for complex user-created artefacts such as the derived productivity table to become joint assets and evolve over time with multiple contributors
- its data model revisited in the context of the IDI redesign
- more methodological research and practical guidance on analytical issues specific to using weighted longitudinal data that has been matched/linked beyond what was envisaged in the original design (this is also an issue to help with the IDI)

Some of these are already planned or under way.

The priorities for the IDI from a user perspective are:

- a researcher-oriented guide or induction manual, and possibly training and buddy system
- a way to share functionality for key repeat tasks rather than just code snippets
- better ways to share and jointly develop code, faciliated by version control
- a possible “population explorer” layer and tool, which I will discuss in my options paper on layers
- more of the data joining, re-coding and filtering code written in SQL and executed on the database server, reducing data movement on the network, saving more data in the database designed for it, and using the database server as a data crunching tool not just a static pool of data
- revisit the data model during the “Fundamental Redesign”
- tidying up of the way metadata is treated so it is better linked to the database and manages change over time
- tidying up the presentation of the SIAL, and addition of a few further useful tables and views such as the MBIE “Spells” data tables and, when available, data on relationships, households, spells as a NEET, and other possibilities.

In terms of sequencing, version control is the priority. Version control software, properly rolled out and introduced to researchers, should lead to a radical change in the scalability, reproducibility and flexibility of code development and sharing. The shape of the problems and solutions outlined above will vary as researcher practice adapts to the availability of version control (for example, joint functionality might evolve more spontaneously when the version control tools to facilitate it are in place).

2 Purpose and approach

2.1 Background

This document sets out to understand the technical practicalities of how researchers currently approach analysis of integrated data in the Stats NZ Data Lab environment. The primary motivation is to inform an options paper in September 2017 on approach to design of information and analytical layers for the Integrated Data Infrastructure (IDI). Deciding on and if appropriate and possible building such layers is one of the four workstreams of the “IDI 2” project.

The aim is to represent the state of IDI and LBD research in August 2017 and analysis is based on data collected at that time, which will rapidly get out of date. This report does not aim to be an introduction manual although it might provide useful background reading for IDI and LBD researchers. As a working paper in a broader project, it assumes a fair degree of background knowledge and does not attempt to fully define the context and many issues that are taken for granted from the readers.

The report represents the views of the author and should not be attributed to Stats NZ. All responsibility for errors and omissions (which certainly are present) lies with me. Due to time constraints I wasn't able to conduct anywhere near as much research - both interviews and investigations repeating researchers' analysis - as I would have liked. Comments and suggestions are welcomed.

2.1.1 Things to watch out for

The report is released as an HTML document and should be opened with a modern browser such as Chrome, Edge or Firefox for all the imagery to show and for the menu in the sidebar to operate. This report was written using R Markdown (http://rmarkdown.rstudio.com/) and the source code of the report and its analysis is in the Stats NZ document management system (https://stats.cohesion.net.nz/Sites/CR/CRPRS/IDI/ManagementandAdministration/Forms/All%20Documents%20FY.aspx?RootFolder=%2FSites%2FCR%2FCRPRS%2FIDI%2FManagementandAdministration%2FIDI%2Dlayers%2Fhow%2Dis%2DIDI%2Dused&).

In this report, unattributed quotations are from IDI or LBD researchers except where otherwise indicated and are shown like this:

> *"I just love the IDI!"*

I sometimes use snippets of computer code to provide precision and remind us all that we are talking about a relational database after all. Unless otherwise indicated by the context, code is in Structured Query Language (SQL) and looks like this:

```
SELECT TOP 100 * FROM IDI_Clean.data.personal_detail;
```

When I use the exact name for databases, schemas tables and servers it looks like this:

The `IDI_Clean` database sits on the `wprdsql36` server.

2.2 Research questions

The questions I set out to address can be categorised as follows:

2.2.1 What is the integrated data and analytical platform available to researchers and how is it presented to them?

Early into the exercise I realised that there was no single researcher-oriented artefact that explains in sufficient technical detail the full range of the data, research environment and analytical platforms that make up New Zealand's integrated data. So the first part of this report sets out to describe, from a researcher's point of view, what data is available, how it is structured, and the analytical platforms that are available. This description also forms a starting point for understanding researchers' hopes and frustrations, and opportunities in the system that services them.

2.2.2 What are researchers trying to achieve?

My own aims are to use this work to help assess the potential of future "layers", which might meet the need of users who currently do not use the integrated data. However, to identify any such potential I needed to know what is currently *possible* and the best starting point for this is what researchers are *trying to achieve* and what they are *actually achieving*. Hence I investigated as diverse a range of research projects as I could in the time available to form a broad view of what was being done and what is possible with the integrated data.

2.2.3 What toolset is being used by researchers and how are they using them?

To understand opportunities, I needed to understand the way researchers currently use the tools available to them. Hence I spent some time analysing and reviewing coding practices, how folders are structured, and how people appear to manage their workflow across the various tools.

2.2.4 How do researchers work together?

As my project is considering the potential role of "layers" as part of the research infrastructure I have a particular interest in how researchers create and use joint assets. A layer would be such a joint asset itself. The researchers currently build and maintain assets in the form of shared code and other tools and knowledge both within their projects and across projects. I am interested in how they work together to build a community and corpus of knowledge, jointly develop the code base, and introduce new analysts to their broader and project-specific communities.

2.2.5 What opportunities exist to make life easier for researchers, and to capitalise on the gains they've made so far?

While I have a particular focus on the role of layers and they will be the subject of a coming options paper, I anticipated coming across a range of additional opportunities for improvement that could be passed to the other workstreams of IDI 2, particular those involved with "Access Pathways" and the "Fundamental Redesign". So this report aims to identify such areas of potential improvement.

2.3 Defining terms and "what would good look like"

This isn't an evaluation but we still need a framework with normative elements of good practice for answering the questions above in a way that makes sense for the reader as well as myself. We also need to seek some common understanding on terminology.

2.3.1 Databases

This report is about the databases that make up the IDI and the LBD. But many readers are probably interested in databases only as end users and may not be familiar with the sometimes confusing terminology. It is worth while setting out how we will be using some terms.

Term	Description

relational database management system (RDBMS) (http://searchsqlserver.techtarget.com/definition/relational-database)	A collection of data organised into formally-described tables from which it can be accessed or reassembled in many different ways without having to reorganize the tables. The tables *relate* to each other by means of variables in common such as `snz_uid` in the IDI which appears in many tables as an individual's confidentialised identification number, assigned to them at random during the quarterly re-build of the IDI. Other types of databases exist, with much attention recently on graph databases (https://en.wikipedia.org/wiki/Graph_database) and other types that are commonly part of the "big data" world and handle less structured data in ways that relational databases cannot. However, the data in the IDI and LBD is fundamentally relational in the way researchers think of it and use it; even if it were moved into a "big data" graph database it would probably still need to be presented to researchers in relational form. There are tools that do this (https://en.wikipedia.org/wiki/Apache_Hive).
SQL Server	Microsoft software used by Stats NZ that manages relational databases and comes with applications for developing, managing and tuning databases, building "extract-transform-load" packages for data sources, integrating data, and developing "reports" (ie standard analysis and presentation) from it. For researchers, interaction with SQL Server is usually either via the "SQL Server Management Studio" application, or indirectly via a statistical application that talks to the database under the hood via ODBC (Open Database Connectivity).
server	A computer (for the integrated data, a virtual machine rather than a physical one) that sits in a central location and provides services to multiple other computers. For our purposes, those services are one of: • holding and analysing data in a relational database management system (SQL Server); • analysing data as instructed in statistical programming language (SAS or R); • saving and retrieving files to disk such as programming scripts, data that *isn't* in a relational database, and analytical output (Windows file server).
table	Tables are the basic element of a relational database. They have rows and columns and can be visualised as each being a clean, tidy single spreadsheet. Unlike a spreadsheet there are strong constraints on what can be in each column of a database table eg "only character strings up to a certain length", "only integers up to a certain size", "only TRUE or FALSE".
database	A particular collection of tables. While we often speak of the "IDI database"", it is actually several databases. A single server with SQL Server on it will typically be hosting several databases.
schema	In SQL Server terminology (which differs from other particular database applications), a schema is a collection of tables and views within a particular database. Any particular database will typically contain several schemas. The sequence `server -> multiple databases -> lots of schemas -> thousands of tables` is a strict nested hierarchy ie a table can only be in one schema, a schema can only be in one database, and a database has to reside on a single server.
query	An instruction to the database to return data from one or more tables. A rectangle of data is returned from a query and might be then used (for example) for presentation, statistical analysis, or saved as a table back in the database for future use.
SQL (https://en.wikipedia.org/wiki/SQL)	Structured Query Language, the language in which queries are written to combine, filter, aggregate and analyse data from relational databases. SQL refers to the hierarchy of objects in the database separated by `.` ie following the pattern `database.schema.table.column`. In addition to simple pulling of data, SQL has computer programming functionality such as the ability to repeat actions (via the `WHILE` command) and define variables for repeat use and other manipulation. SQL is also used for database development and administration tasks such as creating tables, views and indexes, changing user permissions, and reporting on performance. SQL is a *language* (with various dialects in the form of its implementation) and should not be used as shorthand for SQL Server which is a particular database *software*. Like other commercial providers, SQL Server's implementation of SQL goes beyond the standard SQL, for example by allowing the development of stored procedures (https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/create-a-stored-procedure) which are computer programs that run a set of SQL for given parameters when executed by a user with appropriate permissions.
index (https://en.wikipedia.org/wiki/Database_index)	A copy of selected columns of data from a table that can be searched very efficiently. Database administrators add indexes to tables to speed up queries. The effect can be dramatic eg 100x or 1000x faster. Analysts don't need to know of the existence of indexes, although they can sometimes re-write queries to make optimal use of the indexes that there are. Indexes can be clustered (which means the entire table is re-written in the order of the index, to make retrieval of associated items even faster) or non-clustered. SQL Server also provides column store indexes (https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview) for use with databases that are primarily for analytical queries (ie like the IDI, and as opposed to managing lots of transactions like the database behind an ATM) and can have even further spectacular performance improvements
view	A commonly used query can be used to define a *view*, which doesn't copy the data but simply makes the query continuously available so analysts don't need to respecify the query. Accessing a view is basically shorthand for the query that defines the view, so if that query is expensive (ie slow) the view will be much slower to return data than if it were run once and the results materialised as a table (ie saved to disk on the server). To help address this, SQL Server has a capability for indexed views (https://docs.microsoft.com/en-us/sql/relational-databases/views/create-indexed-views) which are much faster than a simple view.

primary key (https://www.techopedia.com/definition/5547/primary-key)	A special relational database table column (or combination of columns) designated to uniquely identify all table records. A primary key must contain a unique value for each row of data and it cannot contain null values. The primary key is always indexed (https://stackoverflow.com/questions/10658500/primary-key-is-always-indexed-in-sql-server), so a table with a primary key has at least one index to help speed up queries.
heap (https://docs.microsoft.com/en-us/sql/relational-databases/indexes/heaps-tables-without-clustered-indexes)	SQL Server terminology for a table with no primary key or other clustered index that sorts the table in order on the disk. This means any query of the data has to look at the entire table, and performance is slow. Microsoft advise avoiding heaps when the data is frequently sorted, grouped together, queried on the basis of ranges, or tables are large.
data model (https://en.wikipedia.org/wiki/Data_model)	An abstract formalisation of the objects and relationships represented by data. Not related to statistical or economic models; the data model is a fundamental part of a database design.
normalisation (https://en.wikipedia.org/wiki/Database_normalization)	The process of simplifying data structures to reduce data redundancy and enforce integrity. With the sort of data under discussion here, this often involves reshaping data from a single wide table where the column names indicate something about the data into multiple narrower and longer tables with less NULL values. Fully normalised data in "third normal form" is similar to the statisticians' concept of tidy data (http://vita.had.co.nz/papers/tidy-data.html), defined as "each variable forms a column, each observation forms a row, each type of observational unit forms a table". Tidy data is easy to manipulate, model and visualise. Normalised data lends itself to a more flexible and full range of query processing. However, analytical databases do not necessarily have to be fully normalised (https://www.quora.com/Is-data-warehouse-normalized-or-denormalized-Why) - there are arguments for presenting denormalized layers to analysts for both understandability and performance (for example, to avoid expensive joins of normalised tables in a snowflake schema (https://en.wikipedia.org/wiki/Snowflake_schema)).
data warehouse (https://en.wikipedia.org/wiki/Data_warehouse)	A central repository of integrated data from one or more sources, including a staging area, storage, and datamarts for presenting to users.
datamart (https://en.wikipedia.org/wiki/Data_mart)	The access layer of the data warehouse environment that is used to get data out to the users in a particular department. A datamart might consist of one or more databases designed to meet a particular set of users' analytical and reporting needs.
dimensional modelling (https://en.wikipedia.org/wiki/Dimensional_modeling)	An approach to specifying the data model for a datamart. It divides data into facts such as "person X purchased pharmaceutical A today" and dimensions such as a table of reference data on persons X, Y and Z and another table of reference data on pharmaceuticals A, B, and C. Dimensional models are typically not fully normalised but contain some amount of redundant information or inefficient storage in order to make them more understandable to analysts and reduce the number of joins needed between tables.

Terminology can be confusing! The other use of "schema" outside the SQL Server context is to mean "diagram showing the data model behind the database design". For SQL Server, it is easier to think of schemas as forming a similar function to subfolders within the individual database. Even more confusing, the terms *normalised*, *model* and *dimension* each mean completely different things in the database world to the the world of statistics and mathematics.

Sensible people can disagree about the detail of the best way to design and build databases to support analysis (see Annex F for more discussion in the context of the ongoing debate about data warehouse architecture) but there are some principles on which I would argue there is a broad consensus. These include:

- The database design should be stable and robust to change. For example, to the degree possible, changes in questionnaires in the data collection process, or in a way classifications are rolled up into larger categories, should not require changes in the data model and the queries that run on it, but just to the data inside tables. It needs to be able to handle slowly changing dimensions (https://en.wikipedia.org/wiki/Slowly_changing_dimension), such as (in our context) the definition of meshblocks, or changes to Inland Revenue forms outside of Stats NZ's control.
    - The need for robustness will often require the design of the underlying tables to be fairly abstract. That is, the design should minimise dependence on the specific data to be analysed, but using more general categories such as "survey", "question", and "answer". This leads to a range of benefits including a much more stable structure over time, easier to expand the scope of the database, and easier for analysts to understand the complete scope of the data.
- While the underlying design might be tight and minimal, analysts should be presented with as many versions of the data as they need for convenient analysis, and it should be fairly easy for them to develop or commission such new versions. These "versions" might be views within a database, or they might be entire datamarts that are fed from a data warehouse - the principle is that even if the data is fully normalised in some location, analysts can see the version convenient to them.
- The database should be tuned for performance. There are many subtleties in this but as a basic starting point each table should have a primary key and clustered index that is as narrow as possible. Columns that are commonly used in queries to join tables together, for filtering, or to aggregate data should have indexes. Other indexes should be experimented with in the light of what queries are common. Query performance should be an important criterion in deciding on the degree of normalisation in the data model.
- The database should facilitate all the common data manipulations needed. For example, it should be straightforward to join two tables based on the date column in each table.
- Analysis should be close to the data, and the system should minimise the amount of data movement through the narrow pipe of networks and disk reading and writing in analysts' real time.

2.3.2 Research practices

Good quality research practices should:

- be based on an understanding of the literature and current knowledge of the question

- use statistical methods that are modern good practice
- only make inferences that are justified by the analysis, and which have uncertainty associated with them quantified to the degree possible eg through use of confidence or credibility intervals
- be easily reproducible (ie get exact results with the same data) and replicable (ie get similar results with new data)
- use tools and approaches that help the researcher move quickly through exploratory, analytical and production phases of their work
- use the power of graphics at all points.

2.3.3 Analytical coding

Serious data analysis with data in a relational database requires writing computer code in SQL plus a language for statistical modelling such as SAS, R, Stata or Python. Organising code well is a critical part of organising an analytics project well. An example of best practice for the overall approach to analytical projects is the Microsoft Team Data Science Process (https://blogs.technet.microsoft.com/machinelearning/2016/10/11/introducing-the-team-data-science-process-from-microsoft/).

Some of the characteristics of good practice coding for analytical projects include:

- folder systems are organised to keep inputs, outputs, and analytical code compartmentalised or at least easy to identify; and facilitate work by teams on projects (eg rather than significant development being done in personal folders for each analyst)
- source code version control via a tool such as Git (https://git-scm.com/) or SubVersion (https://subversion.apache.org/)
- bugs, issues, and "TODO" items are tracked systematically
- easy to find documentation that explains both the overview of any analytical project or task, and (with inline comments in programs) the details of its execution
- comments in code that explain the *why* rather than the *what*
- computer programs are broken into a number of modules to make them easier to develop, understand and maintain; and for teams to work on a project simultaneously
- code has plenty of white space and good use of indentation to indicate hierarchical structure - and in general readability is taken seriously.
- work is designed to be easily picked up by other team members, whether as joint development or for peer review and quality control
- team members use a common style guide for their code and scripts are team products rather than idiosyncratic
- repeat tasks are abstracted into macros or functions controlled by analyst-settable parameters, and in general the DRY ("Don't Repeat Yourself") principle is observed
- magic constants (https://en.wikipedia.org/wiki/Magic_number_(programming)#Unnamed_numerical_constants) are minimised and replaced with variables that are clearly declared and explained at the front of programs
- scripts and functions that are going to be run repeatedly should be refactored (https://en.wikipedia.org/wiki/Code_refactoring) for speed
- scripts that are going to be frequently adapted or updated should be refactored for understandability
- any intermediate data objects have a clear provenance and can be easily re-created from scratch unless they were sourced externally
- projects should be portable (eg so they can be shared, or migrated to new platforms) and as context aware as possible. For example, it is better for the program to find out its own location than to hard key into the script absolute file paths; better for it to check the date and add it to an object than to have the date coded in the script, etc.

> *"Programs must be written for people to read, and only incidentally for machines to execute." (Donald Knuth (https://mitpress.mit.edu/sicp/full-text/book/book-Z-H-7.html)).*

2.3.4 Analytical community

A thriving analytical community would be one that embraces a social coding (http://whatis.techtarget.com/definition/social-coding) approach and if successful the following would be observed:

- Questions and answers would accumulate to capture learning and allow collective knowledge to evolve
- Wikis and other user-written documentation would accumulate and be used
- Code would be easy to either share or keep private at the click of a button, and would in fact be shared
- It would be easy to share (with others who have the appropriate security permissions) an entire project including with a sub-folder system, not just individual scripts and excerpts
- Functionality could be easily packaged and shared in a way that:
  - readily portable
  - easy to maintain and update for all users
  - helpfiles are linked to the functionality, and installing the functionality means the user also gets access to the helpfiles, vignettes, and other documentation
  - release numbers can be tracked

This isn't visionary and futuristic - these exact characteristics are standard in the open source software and analytics world that researchers are familiar with, as seen by:

- Q&A sites like Stack Exchange (https://stackexchange.com/) and Quora (https://www.quora.com/). Stack Exchange in particular provides an extraordinary resource on both statistical modelling and inference (https://stats.stackexchange.com/) and SQL, R and Stata (https://stackoverflow.com/).
- Code sharing and distributed integrated development sites like GitHub (https://github.com/) and GitLab (https://about.gitlab.com/), both of which use Git version control software
- Jointly maintained sites such as Wikipedia and the wiki documentation feature on GitHub
- the Stata and R ecosystems, where user-contributed libraries or packages add major enhancements to the base application and combine documentation with portable functionality.

The software behind these developments in collaborative building of knowledge and software is available.

I turn now to answering my five research questions.

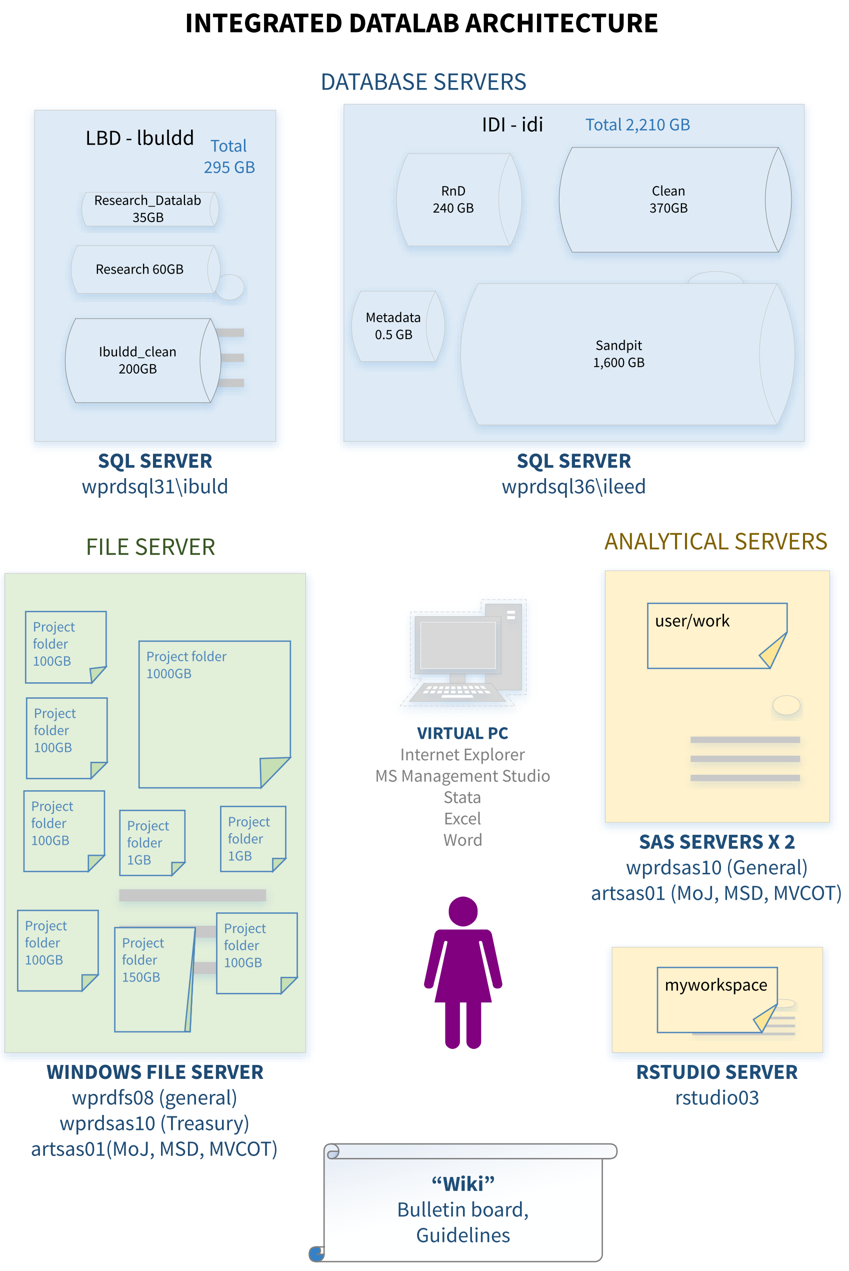
3 The Stats NZ integrated data

3.1 The Data Lab environment

The Stats NZ “Data Lab” hosts the Integrated Data Infrastructure (IDI) and Longitudinal Business Database (LBD), as well as a range of stand-alone microdata datasets. The two main integrated data collections include:

- individual level data on the SQL Server machine with the name `wprdsql36\ileed` :
  - the core of the “IDI” itself in a database `IDI_Clean` of about 370GB
  - `IDI_Metadata` , a separate database around 0.5GB
  - the `IDI_Sandpit` database of around 1,700GB
  - the “Research database”, `IDI_RnD` of around 240GB
- firm level data on the SQL Server machine `wprdsql31\ibuldd` :
  - the LBD proper in `ibuldd_clean` database, around 200GB
  - `ibuldd_research` data is around 60GB
  - `ibuldd_research_Data Lab` , around 35GB

The overall environment is set out, from a researcher’s perspective, in the diagram below. See Annex B for a more formal description of the architecture.



All the servers are virtual machines, housed by Revera Cloud Services (https://www.revera.co.nz/). `artsas01` , the special SAS server dedicated to computationally intensive actuarial modelling under two government research projects, was set up at Revera from its start. `wprdsas10` (the other SAS server) and `wprdfs08` (the file server) were moved to the Revera cloud after the Kaikoura earthquake in November 2016.

Until late 2014, the analysts using `wprdsas10` would save their SAS scripts, data and outputs directly onto that same server. From late 2014, `wprdsas10` users other than Treasury have been saving data to the separate file server `wprdfs08` rather than to the SAS server. The amount of data saved on `wprdfs08` has increased significantly over time, an issue that will be discussed later in the report. Treasury, MoJ, MSD and MVCOT researchers don't have to use `wprdfs08` as their file server but can save files directly to the same machine that is serving SAS (either `wprdsas10` or `artsas01`), and this has important performance implications for them in reducing network traffic between SAS and the file server.

3.2 Introducing the IDI

3.2.1 IDI_Clean

3.2.1.1 Description

The `IDI_Clean` database has 34 schemas and 370 tables. The schemas have names like `acc_clean` , `acm_clean` and `ird_clean` (for data from ACC, Auckland City Mission, and IRD respectively) and nearly all of the schemas each hold the data from a particular data sourc such as Census, the Police, Ministry of Justice, Ministry of Social Development, etc. This use of a schema for each data source rather than a more integrated model is a key design decision for the IDI and came about largely because of the need to allocate researchers' permissions by data source.

Many of the tables have a column `snz_uid` which is the unique identifier for one of the 9+ million individual people represented by data in the IDI. Following the Stats NZ "five-safes" approach to information security, IDI researchers only get access to the schemas they have applied for under their project and been granted permissions.

There are no views provided in `IDI_Clean` , only the underlying tables.

3.2.1.1.1 The `data` schema

The `data` schema is a little different from those named after a data provider, and holds the core information relating to individuals that is available to all IDI users. In many ways, it already functions as start for the "analytical layer" I am to prepare options for in the next paper. Key tables in the `data` schema include:

- `data.personal_detail` has one row per `snz_uid` and is particularly important because it includes a column with a 1 or 0 indicator for whether the individual is part of the "spine" of the IDI. Most researchers want to limit their analysis to individuals who have been linked to the spine. There are over 60 million values of `snz_uid` in the IDI (and hence in `data.personal_detail`), of which only around 9.6 million are linked to the spine. The remainder of the `snz_uid` values represent records from various datasets that could not be linked, meaning many "people" with their own `snz_uid` in this table are duplicate records. `data.personal_detail` is about 1GB in size.
- `data.snz_res_pop` was originally a side product of the Census Transformation project and contains individuals who should be considered part of New Zealand's resident population for a given as at 30 June of an indicated year. There is one row for each unique combination of `snz_uid` and year. There are only 5.6 million distinct values of `snz_uid` in `data.snz_res_pop` - so some individuals in the spine have never been residents on 30 June. Some (but not all) research purposes focus on just individuals who are resident on a particular date. Note that the resident population estimated from the IDI is about two percent more than that published as an official statistic. `data.snz_res_pop` is about 500MB in size.
- `data.source_ranked_ethnicity` is another side product of Census Transformation and it gives the best estimate (on the basis of business rules) of ethnicity for each individual and the source for ethnicity for that individual (Census, Ministry of Health, etc). This saves researchers constructing such a table themselves from the various sources of ethnicity data. There is one row per `snz_uid` , and approximately 10 million rows including 7 million that are attached to the spine (note that this means that not all values of `snz_uid` in the spine have an ethnicity value).
- `data.income_tax_yr_summary` and `data.income_cal_yr_summary` provide rolled up information on income per year for each combination of `year` and `snz_uid` , with a column for each month's income. Other tables in `data` have more detail if needed. 5.4 million values of `snz_uid` have some income information; this includes a small number (about 90,000) that are not linked to the spine.
- `data.person_overseas_spell` contains "spell" data (ie starting and ending dates of a spell overseas for an individual) that is apparently consolidated from the `cus_clean.journey` Customs and `dol_clean.movements` from MBIE, but the only value of `pos_source_code` (ie source for information on person's overseas spell) is "DOL_M" (ie Department of Labour, MBIE's predecessor).
- `data.address_notification` (and `address_notification_full`) contain point-in-time information on addresses. Each row is a combination of a "notification" (eg a citizen telling a government department their address) and an id for the address. Researchers cannot see actual addresses. Extra columns provide information on post code, region, Territorial Authority and meshblock. There are 1.9 million distinct address IDs. There are 100 million notifications in `address_notification_full` and 28 million in `address_notification` . A page on spatial information in the Data Lab "Wiki" clarifies that `address_notification` provides a prioritised address history for all `snz_uid` individuals where address information exists.

> *"What is great about the address notification tables is that StatsNZ has pre-processed the data from multiple sources. From memory, I think they collate address information from up to 8 sources, and then have a priority hierarchy for address changes. This makes our analysis so much simpler. For example recently I wanted to understand the housing instability for a certain cohort… Since this table has already pre-processed the address information from these multiple sources, I could simply count how many (reported) address changes there had been for each person in my cohort in the reference period. Without this table, I would have had to analyse across all the sources (some of which we are likely to not have access to in our project)."*

> *"Didn't have unified ethnicity table at beginning of this project so used census 2013 instead. That was no good for people who were out of the country on census night. Since then there is a new unified ethnicity table which solves a lot of problems – like the tax summary table on income streams, another really useful thing."*

While some of the most important tables in the `data` schema are available to all researchers, others are only available with special permission (eg the exact `data.full_birth_date` is normally kept hidden from researchers to minimise risk of re-identification, but can be made visible for analysis with a specific justification such as research into neo-natal deaths).

3.2.1.1.2 The security schema

The security schema holds a range of information related to the linkage process of building the IDI which is sometimes used by researchers. For example, `security.concordance` contains a table of around 65 million rows (one for each value of `snz_uid` , whether or not they are matched to the spine) and the corresponding value of other ID variables eg `snz_moe_uid` , `snz_dol_uid` , etc. Most of the values are `NULL` indicating that there is no data for that `snz_uid` from the particular data source (Ministry of Education or Department of Labour, now MBIE, respectively). `security.concordance` is used by some researchers to filter their population of interest to only those with data available from the specified data sources.

An SQL query like:

```sql
SELECT TOP 100 *
FROM IDI_Clean.security.concordance
WHERE snz_spine_uid IS NULL AND snz_moe_uid IS NOT NULL
```

returns examples of individuals that are not on the "spine" of the IDI, but do have data from Ministry of Education (and, in most cases, from other data sources such as IRD). My understanding is that there are differing views in the community of researchers about whether such individuals should be included in analysis that joins up across data sources (as opposed to, for example, analysing the Ministry of Education schema by itself, in which case there is no real need to link to the spine). There are definitely risks associated with including these non-spine individuals, particularly across three or more schemas, because the same person could have multiple identities in such data once joined together. In particular, data that has not been linked to the spine may properly belong to an `snz_uid` that *is* on the spine. Researchers' judgement, in light of their overall strategy, is needed to decide whether the risk of including individuals twice (with partial data each time) is better or worse than not including them at all.

3.2.1.1.3 Surveys and administrative data

Seven of the schemas in `IDI_Clean` are related to household or individual surveys. The database is not designed to treat these all as part of a general survey data model, but build a new data model for each survey. The number of tables per survey varies greatly:

Schema	Data source	Number of tables
`ms_clean`	Migrant Survey	1
`gss_clean`	General Social Survey	4
`hlfs_clean`	Household Labour Force Survey	4
`hes_clean`	Household Economic Survey	7
`cen_clean`	the 2013 Census	9
`sofie_clean`	the Survey of Family, Income and Employment	23
`lisnz_clean`	the Longitudinal Immigration Survey	93

The non-survey (ie administrative) data ranges in complexity from just two tables each for Working for Families, New Zealand Transport Agency, Justice, Accident Compensation Corporation, and Childrens' Action Plan; to Ministry of Education (16 tables) Child Youth and Family (17 tables), Ministry of Social Development (21), Ministry of Health (23) and Student Loans and Allowances (26).

3.2.1.1.4 Size of tables

A table of every column of the IDI and information on its table and schema has 8,890 entries and is invaluable for analysts seeking to understand it. This probably should be published as an Excel or csv document. Those with access to the IDI can generate this themselves with (in SQL):

```sql
use IDI_Clean;
select * from information_schema.COLUMNS where table_catalog = 'IDI_Clean';
```

Some of the survey tables are very wide eg `gss_clean.gss_person` has 701 columns, `hlfs_clean.data` has 381 columns and `ms_clean.migrant_survey` has 275 columns. The widest table of administrative data is `dia_clean.births` with 182 columns; the median number of columns is 12 and the mean is 24.

There are nearly 10 billion rows in total; the mean number of rows per table is 25 million and the median is 400,000. The largest tables are `moh_clean.pharmaceutical` with 640 million rows and `dbo.fact_job` at 630 million. `dbo.fact_job` comes from the Linked Employer-Employee Database and is part of the build process for the IDI, not available to researchers. The next three longest tables are also health-related: `lab_claims` (300 million rows), `pho_enrolment` (220 million) and `pub_fund_hosp_discharges_diag` (180 million). So the largest tables available to researchers are all related to health transactions - purchase of pharmaceuticals, accessing laboratory services, enrolment in a Primary Health Organisation, and discharges from a public hospital.

Four of the tables on `IDI_Clean` are larger than 10GB with the largest (`moh_clean.pharmaceutical`) 50GB in size. Most of the individual tables are less than 1GB.

While the IDI is large, it does not qualify as "big data" by itself, particularly not when presented in snapshot form as a single "refresh". While there are many definitions available, I define "big data" as data that is too large and/or unstructured for relational database management systems, collected from digital traces of everyday activities. The IDI is aggregated into a relational structure and is well managed by a traditional relational database management system. SQL Server is used to handle much larger databases effectively and is a perfectly appropriate tool for managing it. There could be advantages to big data tools featuring at some point in their production process, particularly in terms of further rationalising of Stats NZ data pipelines; these are questions for the Fundamental Redesign and in particular its relationship to other data projects in Stats NZ.

3.2.1.1.5 Other things to know

Several of the schemas in `IDI_Clean` are different from the general pattern. For example, the `IDI_Clean.metadata` schema (not to be confused with the `IDI_Metadata` database discussed in the next section) is a special case. It is not available to general researchers as it contains identifying data used by Stats NZ in the linking process.

During each quarterly refresh process, `IDI_Clean` is re-created from scratch. Amongst other things, this means that values of `snz_uid` do not persist across refreshes, a fact that has caught out several researchers and also makes it impossible at this point to permanently attach a random "seed" to individuals as part of an automated confidentialisation process. The `snz_uid` changes on account of both new data and changes that suggest previous links were incorrect; if any redesign is to result in persistent values of `snz_uid` , methods will need to be chosen to deal with such challenges.

As part of the current refresh process, a date-stamped copy is made of the last version of the database `IDI_Clean` eg a new database called `IDI_Clean_20170420` . This means that research can be made either reproducible (by referring to a date-stamped database) or repeatable/updatable (by referring to `IDI_Clean`). As would be expected, these date-stamped databases have increased in size over time as more tables are added and data in existing tables grows; for example, `IDI_Clean_20120402` , the oldest version on the server, is only 140GB compared to `IDI_Clean_20170420` which is 400GB (and hence slightly larger than `IDI_Clean` for some reason).

3.2.2 IDI_Metadata

The small but vital `IDI_Metadata` database, on the same server as `IDI_Clean` , has one schema available to researchers, `clean_read_CLASSIFICATIONS` . It contains 283 tables with classifications that translate codes (such as `post_code` or `sla_Ethnic_code`) into meaningful text.

In a notional mega-star-schema of the IDI, the `IDI_Metadata` database mostly provides the "dimensions" tables, corresponding to the tables in `IDI_Clean` , which mostly contain "facts" when using dimensional modelling terminology. Some of the information in `IDI_Metadata` could be seen as either a dimension or a fact - such as socio-economic deprivation scores by meshblock for 2013 and 2006. The implicit rationale for locating such information in `IDI_Metadata` rather than `IDI_Clean` is that `IDI_Clean` is reserved for information about individuals, and all other information is additional collateral of some sort and hence included in `IDI_Metadata` .

To preserve the full original information from providers, the data in `IDI_Clean` has not been fully standardised during its Extract-Transform-Load, but left representing facts and dimensions as seen by the various data sources. This becomes obvious when inspecting the `IDI_Metadata` database and observing (for example) that there are separate tables of ethnicity classifications in `cyf_ethnicity_code` , `cor_ethnicity_code` , `CEN_ETHNIC05` , `msd_ethnicity_code`, etc. The different sources of the schemas in `IDI_Clean` do not just have different understanding of an individual's ethnicity but also have differing views on what codes to use for which ethnicity). This obviously adds to the complexity faced by researchers, but is a result of wishing to preserve for them the full granularity of original, minimally transformed data.

The tables in `IDI_Metadata` do not use point in time architecture (https://www.outsystems.com/forums/discussion/7040/tip-database-design-point-in-time-architecture-and-soft-deletes/) or other such methods for dealing with change, and the overall system will struggle with slowly changing dimensions. I experience such problems myself in later case studies. Some researchers commented that uncommunicated changes sometimes happened in `IDI_Metadata`. It is almost certainly sub-optimal that `IDI_Metadata` is treated as a separate database to `IDI_Clean` rather than treated as part of a single overall data model that is built together, tied together with foreign key constraints, and tested for consistency.

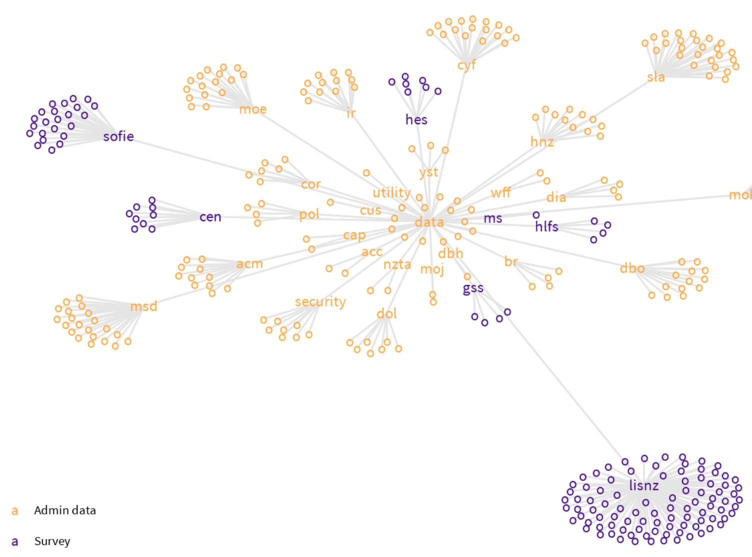
Researchers reported that the metadata, both in `IDI_Metadata` and in other collateral (eg Excel data dictionaries in the "Wiki") is sometimes significantly out of date.

In the early years of the IDI, there was no `IDI_Metadata` database and researchers had to apply variable labels and classifications with the help of SAS code. Clearly the existence of `IDI_Metadata` is a major step forward from this, but researchers' practice has not yet fully caught up. There is a lot of legacy code that applies information now available in `IDI_Metadata` more manually, mostly via the `FORMATS` functionality within SAS; some of this legacy approach is still being used and expanded on today rather than refactored to draw on the `IDI_Metadata` database. Even some of the most experienced researchers are not aware of the `IDI_Metadata` database.

3.2.3 Some reflections on the IDI data model

Schemas and tables in `IDI_Clean`

Each dot represents a table



What makes the IDI a distinctive database management problem is not its size in terms of disk space, but the diversity of the origins of the data, which is passed through directly to the data model.

Taken collectively, the `IDI_Clean` and `IDI_Metadata` databases together resemble a an uneven and incomplete large star schema (https://en.wikipedia.org/wiki/Star_schema) across 35 schemas and two databases. Most of the `IDI_Clean` schemas provide fact tables; and `IDI_Metadata` and some of the tables in `IDI_Clean.data` (eg `data.personal_detail`) provides the dimension tables (ie lookup tables from codes to classifications, and invariant or slowly changing information such as sex and ethnicity). Many of the fact tables are "factless fact tables" that have no column of numeric values but exist just to show the coincidence of various dimensions such as people, starting dates, ending dates, and an event such as going to school, receiving a service, etc.

The interactive table below shows all the column names in `IDI_Clean` that feature in at least two tables.

Show 10 ▾ **entries** **Search:**

	COLUMN_NAME	freq
1	snz_uid	296
2	snz_dol_uid	100
3	lisnz_person_ref_nbr	85
4	snz_ird_uid	46
5	snz_msd_uid	43
6	snz_sofie_uid	24
7	snz_idi_address_register_uid	22
8	snz_moh_uid	21
9	lisnz_occurrence_nbr	20
10	snz_swn_nbr	19

COLUMN_NAME **freq**

Showing 1 to 10 of 281 entries Previous 1 2 3 4 5 … 29 Next

This list of shared column names would be our starting point for seeking to set up an (excessively granular version of) an enterprise data warehouse bus matrix (https://en.wikipedia.org/wiki/Enterprise_bus_matrix) to identify shared or "conformed" dimension variables across the integrated data, if we were following that style of warehousing.

However, the IDI is far from conforming to a dimensional model star schema:

- some crucial dimension information is missing, for example the names of the six standard ethnicity groups and other variables that are embedded in column names of tables
- some standard dimension tables such as a `DATE` table with columns for day of the week, month, national holiday, provincial holiday or not, etc are missing
- columns which should be conforming dimensions are in differing formats that prevent easy joins (eg date is sometimes a DATE variable and sometimes DATETIME)
- slowly changing dimensions are not allowed for
- while some degree of non-normalised data is expected in a star schema for ease of interpretation and minimising joins, there are many instances when more normalisation would be useful, most particularly in the case of the very wide survey data. This issue is even more important for the LBD so is discussed more in that section of the report
- there are discrepancies in the data such as values of `snz_uid` that appear twice in `IDI_Clean.dia_clean.births` ie two records of being born.

There are no foreign key constraints in the IDI. This doesn't concern the end-user as it is a write-once datamart presentation layer; but including foreign key constraints in the build process might help Stats NZ prevent slip-ups described to me where some of the issues above (eg difference between DATE and DATETIME columns) are accidents. More surprising is that there is only a small number of primary keys and indexes and this will be discussed further below. Primary keys and indexes would dramatically improve performance; primary keys would also help identify problems such as duplicate values of `snz_uid` in `IDI_Clean.dia_clean.births` (in fact this is how I noticed it, when making `snz_uid` a primary key in a table I had derived from that one).

The IDI 2 redesign will need to carefully think about the grain of tables and whether the data is sufficently normalised. Most of the administrative tables in `IDI_Clean` are normalised and are very tractable to work with ie easy to write queries with them in combination with the classifications in `IDI_Metadata`. However, my view having worked with the data for this exercise is that there could be advantages from more normalisation:

- Some of the tables are "untidy" (http://vita.had.co.nz/papers/tidy-data.html) in a wide way, with information contained in column names rather than being stretched into long thin fact tables. One impact of this sort of data model is that the metadata in the form of dimension tables cannot be joined directly to the data because facts are encoded in column names, which increases the likelihood that query results will be hand-classified and metadata manually consulted. Examples of wide data include:
    - `dia_clean.births` has columns with names like `dia_bir_parent2_ethnic_grp3_snz_ind`, `dia_bir_parent2_ethnic_grp4_snz_ind` and so on. More normalised data would have a column for value, a column for `ethnic_grp_number` and a column for `parent_number`. In a dimensional model, this table (and some others) would be a candidate for considering as a factless fact table (http://www.jamesserra.com/archive/2011/12/factless-fact-table/) ie it is basically a collection of dimensions (parent, parent, child, date) that coincide together for an event.
- Appendix C has example SQL code that takes the `data.source_ranked_ethnicity` table, normalises it further by making the six columns for ethnicity a individual-category pair, shows how this can make for more efficient combination with the ethnicity classification, and how the original table could be re-created as a database view if wanted.

Researchers trained in statistics and economics are sometimes more familiar with the wide format data, which resembles the final stage of data to be fed into tools that fit statistical models; heavy database users are more likely to seek data that has been further normalised to make it easier to manipulate right up to the last moment when it is "pivoted" (SQL Server terminology) or "spread" (R tidyverse terminology) ready for use in a statistical model.

3.2.4 IDI_Sandpit

The `IDI_Sandpit` has 93 schemas and 1973 tables and views. Datasets on `IDI_Sandpit` fall into four categories:

- schemas owned by individual projects and named after them (eg `DL-MAA2015-51`). The data in these can only be seen by researchers who part of the project, who have full permission to create and destroy tables, views and indexes as part of their data management. There are six such schemas associated with individual Stats NZ researchers and 36 associated with research projects. Projects need to request a schema if they want to use this facility, and most do not have one. These schemas have two purposes:
    - Some researchers use these schemas to store intermediate datasets or views they want to refer to several times, or to do data-intensive processing close to the home of the data to reduce network traffic
    - Stats NZ also used these schemas to make available project-specific ad hoc loads of data, although this role is now taken by `IDIRnD` (see below).
- schemas with names like `clean_read_moe` which are tables that are *either* "ad hoc" loaded rather than part of the regular on-going refresh process, or downstream intermediate tables created by researchers contracted to Stats NZ. These are available to people with permissions to see the relevant data, in this case Ministry of Education.
- staging and processing environments used by Stats NZ, for the refresh process, and only accessible by Stats NZ

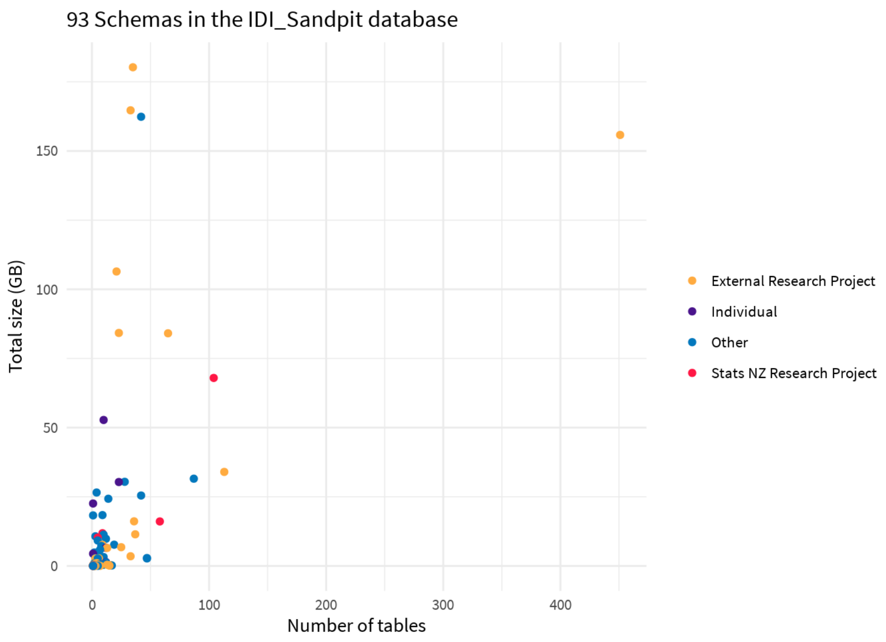
Some of the tables on `IDI_Sandpit` are very large - up to 90 GB (about a quarter the size of the entire IDI, and much larger than any individual table in the IDI). One de facto purpose of the Sandpit has become as a temporary holding place for data that "missed" the latest IDI refresh but is still wanted by researchers seeking up to date or enhanced data. This practice leads to an extremely confusing proliferation of rival versions of the data.

Stats NZ has a policy on researchers' use of the sandpit. The policy sets out the purpose of the sandpit as for creating tables that:

- need to be kept on a semi-permanent basis and take hours to create

- need to be shared between researchers
- are useful for other researchers working in the IDI.

Projects are allowed a terabyte each of disk space on `IDI_Sandpit` but in practice none exceed 200 GB as seen in the chart below:



The policy states that “researchers are granted access to the sandpit schemas that correspond to the data schemas they have access to. For example, if a researcher can see the MoE schema in the IDI, the will be able to see the MoE sandpit schema”. The Policy implies that researchers can create tables and views in the Sandpit schemas they have access to in order to share data with other researchers with appropriate permissions but this is not the case, as seen in the failure of a command such as:

```
-- I have access to this table in the sandpit but can I make a copy of the top of it?
SELECT TOP 1000 *
  INTO IDI_Sandpit.clean_read_CYF.can_i_make_a_table
  FROM IDI_Sandpit.clean_read_CYF.cyf_EV_CLI_CFAS_INVESTGTNS_CYS_F
-- No.
```

In practice, researchers can only make tables and views in their project’s nominated schema, and they are not then visible to other researchers. Only Stats NZ can use the Sandpit to meet its second and third purposes as stated above.

With its multiple purposes and conflicting use, the Sandpit is both confusing and an under-utilized asset.

3.2.5 IDIRnD

`IDIRnD` is a relatively new database on the `wprdsql36` server that, since the June 2017 refresh, Stats NZ uses for:

- loads of individual ad hoc datasets (taking over from the Sandpit)
- intermediate tables and views for use by multiple researchers (currently the only such example is a version of the Social Investment Analytical Layer tables and views, with date stamps added to the names of the tables).

3.3 Introducing the Longitudinal Business Database (LBD)

3.3.1 ibuldd_clean

3.3.1.1 Basic structure

The LBD is much smaller in size than the IDI. The `ibuldd_clean` database is only 200GB in size, much of it different versions of the same data (as the original raw loaded files are in the database as well as the “final” version researchers are expected to generally use). The data is all in a single schema named `dbo` (“database objects”).

Unlike the IDI, the LBD shows signs of being explicitly designed in a traditional star schema of dimensions and facts. There are:

- 5 dimension tables with names like `dbo.dim_firm_size`, `dbo.dim_firm_size_bos`, etc.
- 18 reference tables with names like `dbo.ref_anzsic96` and `dbo.ref_meshblock`.
- 103 fact tables with names like `dbo.fact_lbd_enterprise_year`, `dbo.fact_ir4_enterprise_year` and `dbo.fact_aes_enterprise_year`
- 78 tables with names beginning with `load` which contain “the raw data so that researchers can choose to undo any processing that has occured. For example… while Statistics NZ construct an overseas merchandise trade fact table that includes the aggregate annual exports of each firm, the database also includes the underlying daily shipment level data.” (from Fabling and Sanderson’s *Rough Guide to the LBD* (http://motu-www.motu.org.nz/wpapers/16_03.pdf))
- 6 tables with names beginning with other strings such as `stage`

The `fact` tables come from 17 different sources of which the `bos` (Business Operations Survey) has the most tables (42) followed by `gap` (20) and `rad` (13).

Many of the LBD tables have foreign keys constraints, primary keys and indexes.

Some of the data is out of date eg `gap` (government assistance programs) and `aes` (Annual Enterprise Survey, only has data to March 2014). This reflects resourcing constraints at Stats NZ and elsewhere, and the semi-manual process of updates.

Unlike the IDI, access to the LBD is "all or nothing" - there is no restriction of access to only the minimum part of the database that is required for a particular research project.

3.3.1.2 Size of tables

The `fact` tables in the LBD are considerable wider than those in the IDI, with 322 columns on average and many tables from the Research and Development Survey (`rad`) and the Business Operations Survey (`bos`) with more than 900 columns. The table below shows the number of tables and columns for the different collections of data in the LBD:

type	source	Number of tables	Max columns	Average columns	Median columns
arch		1	2	2	2
dim		5	9	6	6
dtpr		1	7	7	7
fact	aes	1	300	300	300
fact	aps	1	84	84	84
fact	bai	1	24	24	24
fact	bfs	1	407	407	407
fact	bop	3	144	69	49
fact	bos	42	973	614	586
fact	bps	1	367	367	367
fact	cus	8	26	21	21
fact	gap	20	20	11	10
fact	i10	2	86	82	82
fact	inn	1	140	140	140
fact	ipo	1	16	16	16
fact	ir4	1	82	82	82
fact	lbd	1	15	15	15
fact	lbf	1	67	67	67
fact	leed	5	55	16	8
fact	rad	13	975	390	179
load		78	263	70	52
ref		18	23	8	6
stag		3	102	56	64
sysd		1	5	5	5

Corresponding to their relative wide-ness, the `fact` tables are relatively short - a mean of 2.8 million rows and median of 6,300. Most of the LBD tables are sample surveys - there is less administrative data available on firms than there is on people.

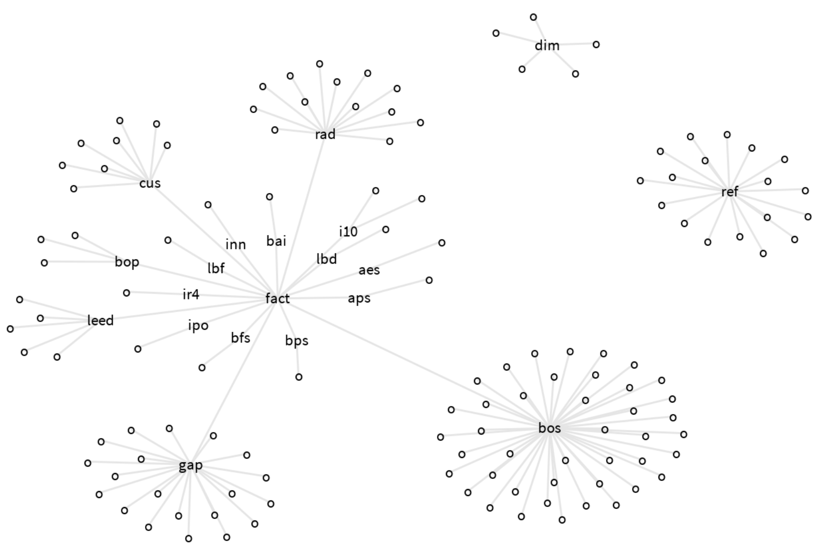
3.3.1.3 Processing

The data in the LBD has had more post-collection processing by Stats NZ than the data in the IDI. For example, missing values have generally been imputed (and the fact of imputation marked with a flag).

Unlike the IDI, the LBD is not created from scratch each time there is new data, but only the new data is updated. This is because the spine of the LBD is the longitudinal business frame, which is maintained as a register. There is no equivalent register of individuals for New Zealand. The fact that the LBD is not re-created from scratch regularly has a number of benefits for researchers, providing a more stable database.

3.3.2 Some reflections on the LBD data model

Conceptual hierarchy of tables in three parts of the `dbo` schema in `IBULDD_Clean`
Each dot represents a table



The LBD database is much closer to a conventional dimensional model star schema than is the IDI and there has been more success in identifying the dimension tables in the `dim` and `ref` parts of the `dbo` schema. Some parts of it, such as the Annual Enterprise Survey and the Longitudinal Business Frame, are very tractable to work with.

The interactive table below shows all the column names in the `dbo` schema of `ibuldd_clean` that feature in at least ten tables. Many variables exist across many tables in the LBD; this is partly a result of it being a tightly integrated database measuring a finite range of things on businesses, but also due to redundancies in the database design (some tables that could be consolidated exist as separate tables):

Show 10 ▼ **entries** **Search:**

	COLUMN_NAME	freq
1	enterprise_nbr	119
2	dim_year_key	105
3	inst_sector_code	53
4	gte_enterprise_nbr	52
5	business_type_code	50
6	response_code	49
7	response_count	49
8	strata_code	46
9	final_weight	44
10	adjusted_weight	43

Showing 1 to 10 of 228 entries Previous 1 2 3 4 5 … 23 Next

However, there are signficant challenges with the shape of some of the other data in the LBD. Broadly speaking, these are known to both Stats NZ and researchers and it is a matter of time and resourcing to improve them.

3.3.2.1 Incomplete consolidation of survey waves

The consolidation of surveys into tables from the original data is incomplete incomplete. For example, while the Annual Enterprise Survey has been successfully (and very usefully for the researcher) combined into a single table:

TABLE_NAME
fact_aes_enterprise_year

… whereas the Research and Development surveys exist in 13 tables:

TABLE_NAME
fact_rad_enterprise_1996
fact_rad_enterprise_1998
fact_rad_enterprise_2000
fact_rad_enterprise_2002

TABLE_NAME
fact_rad_enterprise_2002_uni
fact_rad_enterprise_2004
fact_rad_enterprise_2006
fact_rad_enterprise_2008
fact_rad_enterprise_2010
fact_rad_enterprise_2012_part1
fact_rad_enterprise_2012_part2
fact_rad_enterprise_2014_part1
fact_rad_enterprise_2014_part2

..and the Business Operations Survey has 42 tables (not shown), one for each module of the survey from 2005 to 2016. These 42 tables have not been chosen for analytical convenience or an efficient and tractable data model, but because that is how the data come in. Hence any analysis of the Research and Development Survey or Business Operations Survey that wants to look at more than a single wave of the survey must first join together numerous tables, making a range of difficult judgements on the comparability across time of various questions, and adding many opportunities for coding errors. A SQL program is circulating amongst researchers that does this for the 2005 to 2013 waves, but this is only available if you know the right people.

The Integrated Data team at Stats NZ are well aware of the challenge, which requires time and resourcing to fix.

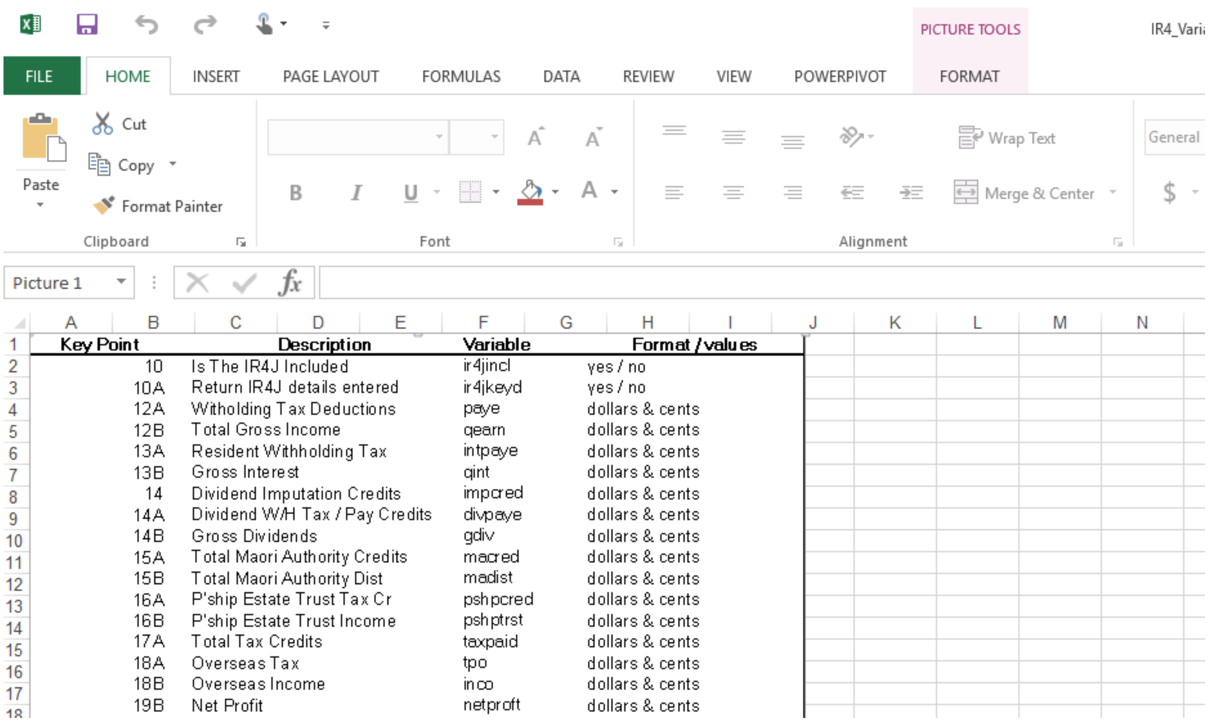
> *"The BOS is disastrous yet it's a really important part of the LBD. Someone should take a hit for everyone else and just create a usable version of the BOS ie with data dictionary, proper questions, proper shape etc."*

3.3.2.2 More normalising desirable?

Like the survey data in the IDI, the data in the LBD is only partially normalised, by design (ie because researchers probably prefer the wide, flattened form familiar from spreadsheet-style applications). The information in the fact tables has one row per response (usually a firm's response to a particular wave of a questionnaire) rather than one row per response - question pair. The one row per response grain is a familiar format for most current researchers (it's how it would appear in SPSS or Stata for example) and matches their expectations and experience but is not an efficient way of storing the data or a particularly tractable form for its manipulation.

For example, the Annual Enterprise Survey tables have 252 columns that finish with `_amt`, indicating an amount of some kind. Identifying columns with this sort of pattern in an SQL query is cumbersome. It requires either manually listing column names, writing a computer program to write the SQL, or unpivoting the data into a long format. This latter approach is much preferred but would normally be part of the Extract-Transform-Load of building the datamart rather than left to the analyst. Alternatively, the whole table can be downloaded to another server where a more flexible language like SAS, R or Stata be used to refer programmatically to the many columns. These alternatives are the reality researchers are used to (and as the data is not particularly large, it is quite feasible to download the entire dataset for one of the surveys into eg the RAM on `rstudio03`) but there is no reason this should always be the case.

Critically, the full text of the questions or sub-questions that columns in the current data model represent is not available as a table in the database. Hence there is no programmatic way to match the thousands of columns to what they actually represent. The information exists in Excel workbooks for each survey, which are not set up to be machine readable; for example, cells are merged, and in at least one case the metadata on what each variable in the IR4 forms means is an *image* of a table, pasted into an Excel workbook:

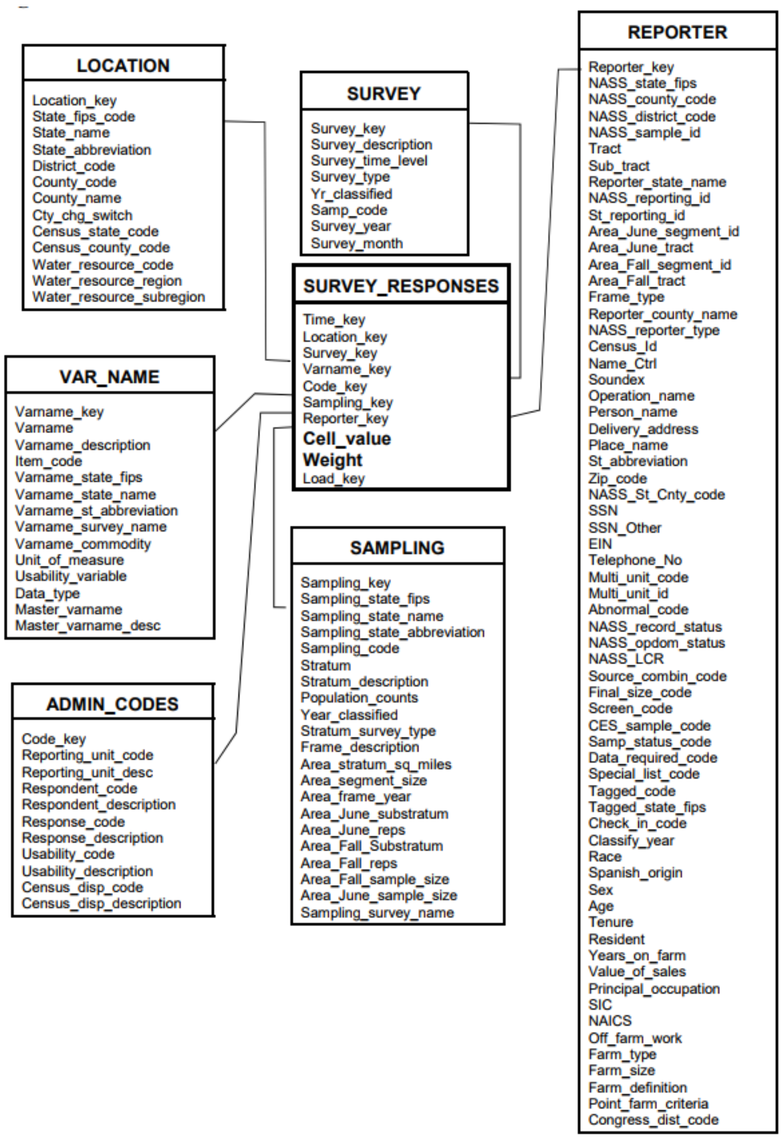


Key Point	Description	Variable	Format / values
10	Is The IR4J Included	ir4jincl	yes / no
10A	Return IR4J details entered	ir4jkeyd	yes / no
12A	Witholding Tax Deductions	paye	dollars & cents
12B	Total Gross Income	qearn	dollars & cents
13A	Resident Withholding Tax	intpaye	dollars & cents
13B	Gross Interest	qint	dollars & cents
14	Dividend Imputation Credits	impcred	dollars & cents
14A	Dividend W/H Tax / Pay Credits	divpaye	dollars & cents
14B	Gross Dividends	gdiv	dollars & cents
15A	Total Maori Authority Credits	macred	dollars & cents
15B	Total Maori Authority Dist	madist	dollars & cents
16A	P'ship Estate Trust Tax Cr	pshpcred	dollars & cents
16B	P'ship Estate Trust Income	pshptrst	dollars & cents
17A	Total Tax Credits	taxpaid	dollars & cents
18A	Overseas Tax	tpo	dollars & cents
18B	Overseas Income	inco	dollars & cents
19B	Net Profit	netproft	dollars & cents

The challenge here also relates to the lack of normalisation; if the data were presented in long thin fact tables with each row representing a survey-time / respondent / sub-question / response, it would be possible to join the fact table to a dimension table with the full text of the sub-question. Views could still be made to represent the data in the more familiar wide, one row per respondent format for analysts more familiar with that structure and to avoid breaking historical code.

The difficulties in linking tables across time are also related in part to the lack of normalisation of the data. If the data were in long thin formats as described above, it would be possible to add additional years of surveys, with different questions as different values in the relevant column, without changing the column names and hence design of the database for each new set of questions.

The problem of a stable and robust single data model for survey data in a warehouse has been solved in principle at least since 1999, when Yost and Nealon (https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/242/2014/05/II-B_Yost_FCSM1999.pdf) published a data model used to integrate all the diverse survey data held by the US Department of Agriculture into a single warehouse. The simple design, with just seven tables, is robust to changing questions from year to year and indeed adding whole new surveys, without additional design and development work:



The Yost and Nealon model uses Kimball's dimensional modelling technique, briefly mentioned in Annex F. The data is *not* fully normalised, but *is* abstracted away from particular survey instruments and questions. The data could be "pivoted" back into wide format (one column per question) if analysts want; in practice such a step should probably be the last one, with a subset of variables, before fitting a statistical model.

The main analytical reason data is needed in wide format is to create the design matrix for a regression or similar model. Leaving it until the last moment in a more tractable form such as that in the diagram above leads to much simpler data manipulations for the analyst. For really large, sparse data there is sometimes no choice - the data has to be moved around in a compact, more normalised format (or other special format for sparse matrices such as SVMlight (http://ellisp.github.io/blog/2017/02/18/svmlite)), and only spread out to be wide when it has reached the server/s that are going to fit the model.

These data model questions are for the IDI 2 redesign to consider.

3.3.3 ibuldd_research

The 60GB `ibuldd_research` serves a similar function for the LBD to one of the functions of the `IDI_Sandpit`, hosting schemas for researchers to create and destroy tables, views and indexes. It has schemas named after individual researchers (not research project number as in the IDI). Data is not visible to other researchers unless special arrangements are made. Only four researchers are using this capability. Researchers are not given `CREATE` permissions to `ibuldd_research` automatically and it may be that other than those four, other researchers are not aware that it is available.alter

Judging from the names of tables in `ibuldd_research`, researchers use it for storing a combination of intermediate data objects (eg `LEED.pent`) and concordances, classifications and reference information not available in the `ibuldd` (eg `currencies`, `country_concordance1999_2011` and `gdppop1999_2010`)

3.3.4 ibuldd_research_database

The 35 GB `ibuldd_research_Data Lab` database holds intermediate data objects for sharing between researchers. It has schemas named after the same four researchers as `ibuldd_research`, but data in the tables is visible to all LBD users.

The `ibuldd_research_database` includes a range of interesting data, such as

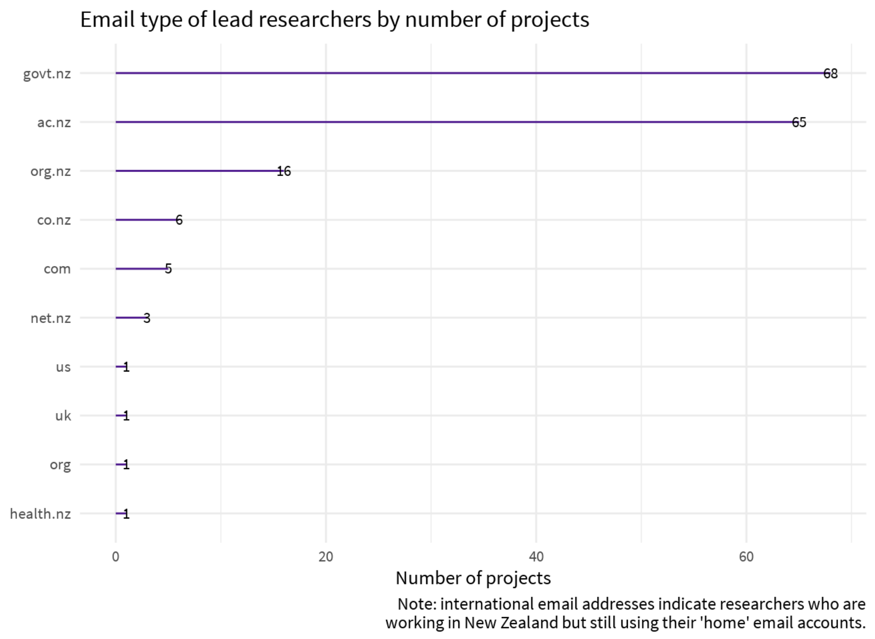
- the meshblocks through which the New Zealand Cycle Trails pass
- a concordance from ANZSIC06 to ISIC Revision IV
- consolidated version of the BOS from 2005 to 2013 into a single table
- a classification of the "slope" of meshblocks into categories A to H (not sure if this is their physical slope or a coefficient in a regression, but I think the former)
- derived productivity by firm (`pent`) and year up to 2014.

This latter table is a shining example of an immensely useful intermediate data object that can be created once and used many times. It presents tidy estimates by firm of FTE, the value of labour, capital and multi factor productivity. If it could be updated whenever new data is added - and there is no reason why this should not be possible - a lot of analysis could take place without joining this table to anything other than classifications. In fact, this one table is so useful it is also a strong candidate to be published as a synthetic unit record file (SURF).

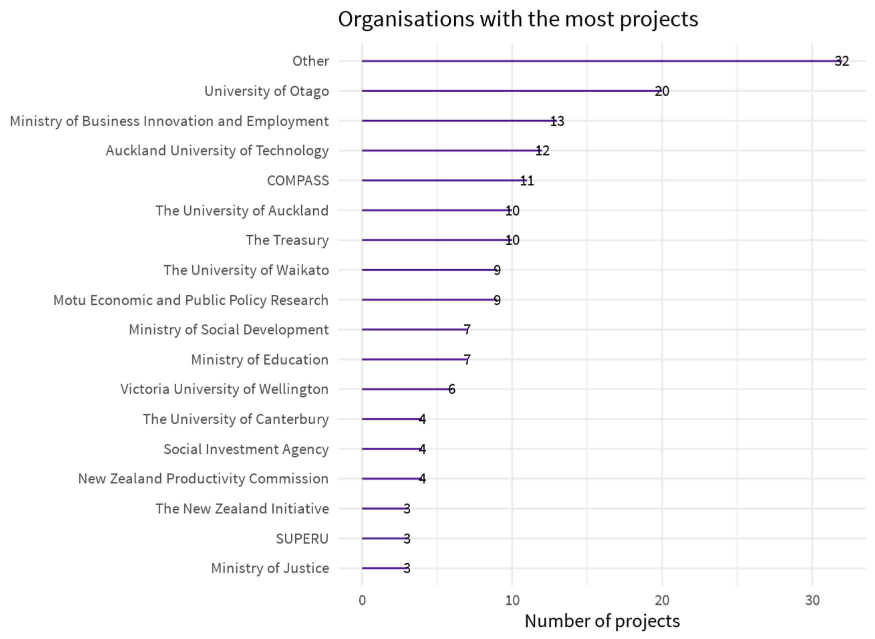
4 What researchers are trying to do

4.1 Where do researchers come from?

We have data in the form of the email addresses of the lead researchers for each of the approximately 170 active IDI and LBD projects. From the chart below we see that government and universities (email addresses that finish with .ac.nz) have roughly equal numbers of projects, with a small number of non-government and commercial groups making up the rest.



Many research projects come from a small number of institutions, as we can see in the next chart:



These figures should be treated as approximate and indicative only, because some entries indicate sub-projects and hence are in effect duplicates.

Nonetheless, we see that universities dominate in terms of numbers of projects for single organisations, which makes sense when we consider that universities are large collections of researchers with disparate interests. Government organisations have more ability to wrap their analytical workstreams into a smaller number of large "projects".

4.2 Four types of analysis, each with the same four steps

The analysis that is undertaken with the IDI and LBD can be categorised into four types:

a. estimating populations (eg the number of people on benefit living in area 'x'), with simple cross tabs and frequency counts

b. research and evaluations with social science or public policy aims to advance knowledge, using methods such as propensity score matching and two stage least squares regression

c. development and deployment of models for purpose of scenario exploration, including by large simulations or simpler aggregate models to inform cost benefit analysis

d. meta-research into the opportunities and limits of administrative data for understanding populations

Many "projects" do two or more of these types of analysis, but most have one type that dominates. Most analysis observed in this exercise was of "type a", and simple cross tabs and frequency counts dominate in the output released through the checking process. Type "b" - chunky research and evaluation projects - is the original archetypal IDI research project around which systems such as researcher applications and output checking have evolved, but is now a (still substantial) minority of the analysis actually undertaken in the Data Lab.

The commencement of the Census Transformation project, some years into the life of the IDI, was probably a watershed moment. Using the IDI to explore population characteristics is quite a different purpose from as a research database, but this use is definitely growing and definitely has potential. However, there are risks involved from this angle, which matter more for characterising populations than they do for research. There is a need for methodology developed to address issues such as bias arising from linkage errors, partial coverage of administrative data, etc. Mostly these issues fall into the work program of the Census Transformation project, but the potential beneficiaries are far wider and include all analysts using the IDI to estimate counts and characteristics of populations.

The four types of analysis feed off each-other. In particular, simple population estimates ("type a") depend on the success of "type d" projects in delivering convenient and reliable ways of inferring to populations from incomplete administrative data. Models for exploring scenarios ("type c") depend on successfully validated social science knowledge ("type b").

4.3 Stated aims of researchers

The stated aims of all approved IDI research projects are available on the Stats NZ website (http://www.stats.govt.nz/browse_for_stats/snapshots-of-nz/integrated-data-infrastructure/researchers-using-idi.aspx#projects). Four examples from that page are excerpted below, one for each of the four types of analysis identified above:

4.3.1 Pilot partnership project: Otago youth not in employment, education, or training (NEET)

> *NEET youth are young people aged 15–24 who are not in employment, education, or training. Stats NZ worked collaboratively with Methodist Mission Southern (MMS) and the Ministry for Women to profile the characteristics and locality of NEET youth, with a particular focus on the Otago region. This project used integrated data to profile the NEET population in MMS's operating area by indicators of vulnerability at relatively small geographical areas. This information will help MMS, and other social service providers, tailor and target their services to young people more effectively and improve outcomes for young people at risk.*

4.3.2 MAA2016-56 Investigating Income Mobility in New Zealand

> *The anticipated outcomes are a more in depth understanding of income mobility which previously could not incorporate the personal and demographic information, particularly household composition, in past research. Researchers contribution is to provide a thorough understanding of income mobility by being able to control for a greater number of variables. The ability of this project's outcomes to contribute to highly topical public debate about inequality, and as in input into key economic and social policy discussions, should render it a good candidate for NZIER's 'public good' support.*

4.3.3 MAA2015-27 IDI-based micro-simulation modelling of the New Zealand tax and welfare system

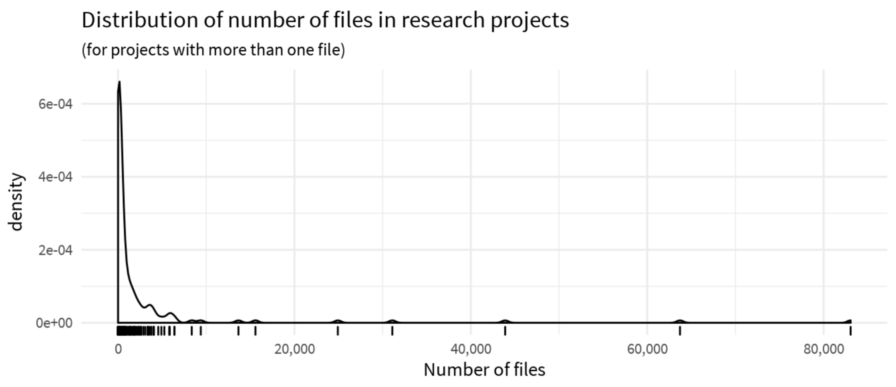
> *The proposed study will investigate using IDI for a micro-simulation model of the tax and welfare systems. Presently, MSD, IRD, and Treasury each use different databases for their specific needs, based on what is available to them. Using IDI allows for multiple datasets to be used in concert, facilitating using a shared model across the agencies. This model should have greater depth and breadth than the individual models, improving the evidence-based advice each agency provides.*

4.3.4 MAA2014-10 Measuring housing affordability in New Zealand as a Tier One statistic

> *This research explores the feasibility to produce a Tier One housing affordability measure by utilising relevant data in the IDI and HLFS, namely, household income, household rents, house sales prices and household demographic profile to form a subset of panel data.*

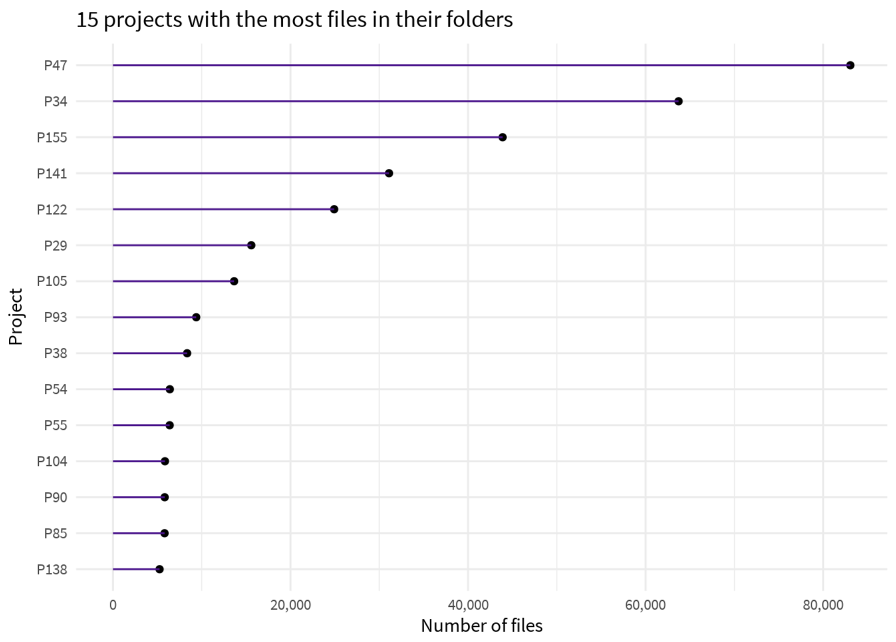
4.4 Overall size of research projects

There are a small number of "mega" projects with tens of thousands of files in their folders, and in general a very skewed distribution even when these outliers are removed. This first chart shows the overall distribution:

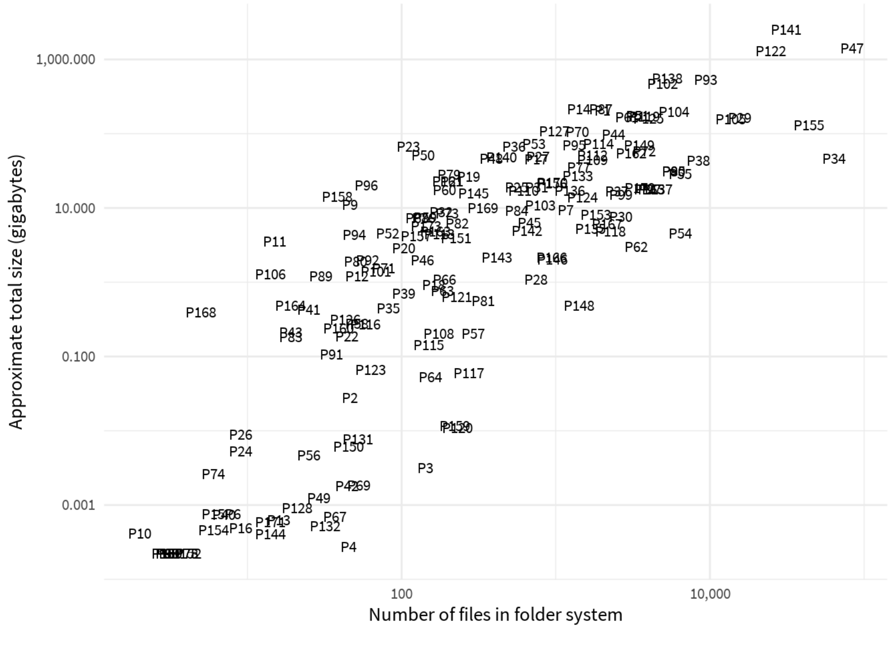


The next charts show the sizes of individual projects. Projects are identified by codes for the purpose of this study.

Five research projects are far larger (in terms of number of files in their folders) than others. Four of these are government department programmes of research that started in 2013 or later; the other is a university programme.



The chart below shows a rough approximation of the actual size of researchers' folders:



Naturally, the projects with more files also take up more space. The largest projects consume more than a terabyte of disk space each; and have tens of thousands of files and thousands of folders. Numbers of sub-folders is not shown here; folder structures for projects are of more interest when we come to qualitative inspection of some projects in more detail.

The table below makes more precise the size in disk space and number of files of the largest research projects currently using Stats NZ's integrated data:

Project	Total Size (GB)	Number of files

P141	2495	31100
P47	1399	83062
P122	1280	24920
P138	554	5254
P93	531	9383
P102	464	4913
P87	214	1962
P14	212	1403
P1	205	2007
P104	195	5858

5 How researchers use the toolkit

5.1 A varying toolkit, folder structure, and coding style

Despite the different aims and levels of complexity of different projects, all research projects looked at for this exercise showed signs of following (and iterating through) a common sequence:

1. exploration
2. preparation of a dataset through filtering, mutating variables and aggregation
3. analysis, ranging from simple counts through to machine learning
4. dissemination, ranging from random rounding a few numbers in Excel through to development of interactive web apps

> *"On a new research topic, most of the time is still preparing the dataset for the specific question – the particular cohort, constructed variables, and their comparison group (if it is that sort of analysis that has a comparison group). All this standardised code has definitely sped things up, but it's a still a job – hours, days or weeks – to get a dataset ready for analysis. There's always quirks about a particular population you're interested in ('has parents or an extended family member like XXX') and constructed variables of interest.""*

The data management process typically includes a combination of three things (not necessarily completed neatly in sequence):

- identifying and filtering a subset of the population of interest
- identifying the range of variables of interest and joining relevant tables together
- rolling up data from "spells" or "events" structure (eg "lived at this address from date X to date Y") to the form needed for analysis, which more typically is in the form of observations that are regularly spaced in time (eg "lived at this region in 2012, this region in 2013, and that region in 2014")

The toolkit used by analysts varies. All projects have SQL at their heart and the first step of any of the projects when the rubber hits the road is a `SELECT...` query. But most analysts are not thinking of SQL as their most important language or developing their SQL in Management Studio. From (estimated) most to least common the approaches taken are:

1. Data management in SAS with a combination of `PROC SQL` (running queries on the database server and extracting results via ODBC) and SAS-native `DATA` steps; analysis is mostly cross tabs done with `PROC FREQ` but occasional more advanced regression and the results are either exported or copy-and-pasted to Excel, where confidentialisation (such as random rounding and cell suppression) is done by hand. A less common variant on this has the confidentialisation done in SAS.
2. As above, with some more advanced statistical modelling done with Stata or R, importing data that was first pulled from the database by SAS.
3. Data extracted with dedicated SQL scripts developed in SQL Server Management Studio and either run in Management Studio (with results exported as some kind of text data file) or directly from Stata or R via ODBC.
4. A variant of the above with the SQL embedded in R scripts.

Of the three statistical languages available, SAS is by far the most commonly used, followed by Stata, with R third. However, most of the SAS code observed for this exercise was doing data extraction, mutation, merging and management tasks or simple cross tabs, rather than statistical modelling and inference.

Projects manage assets in their folders of five broad types:

- input data such as reference information by meshblock sourced from other analysis, or aggregate official statistics like the Consumer Price Index
- code - including three categories of
  - exploratory code,
  - re-usable code such as macros and functions, and
  - final or "production" code for publishable analysis
- intermediate data
- outputs (which is most commonly small or large tables of aggregated data, but can also include images and regression output)
- documentation

There is no standard approach to structuring folders to manage these assets, other than the ubiquitous use of a folder for `checking` and a folder system (with subfolders named by researcher and date) for `checked` - these systems come from the standard output checking and release process. Most larger projects have named folders for individual researchers in a way that enables individual rather than team research, but some are structured around particular workstreams and set up for team work. Most larger projects have a system for sharing assets across the project's researchers such as a common folder to hold key re-usable data in `.sas7bdat` format and a standard set of SAS `LIBNAME` definitions. Best practice projects also have a folder of standard SAS macros to be used in common by individual researchers.

> *"Being sole analyst on a research project (although with subject matter experts) was very tough. Wouldn't do that again. Need someone to bounce ideas against, check code, help trouble shoot. Much better now that I'm working with …, both in the lab at once, even though it's different projects."*

With no version control software available in the Data Lab, the sharing of code, joint development, and management of versioning is extremely primitive and typically done by date stamping different versions of scripts. Serious analytical code development, particularly by teams, is not possible without version control, and the absence of version control manifests in numerous ways in the Data Lab.

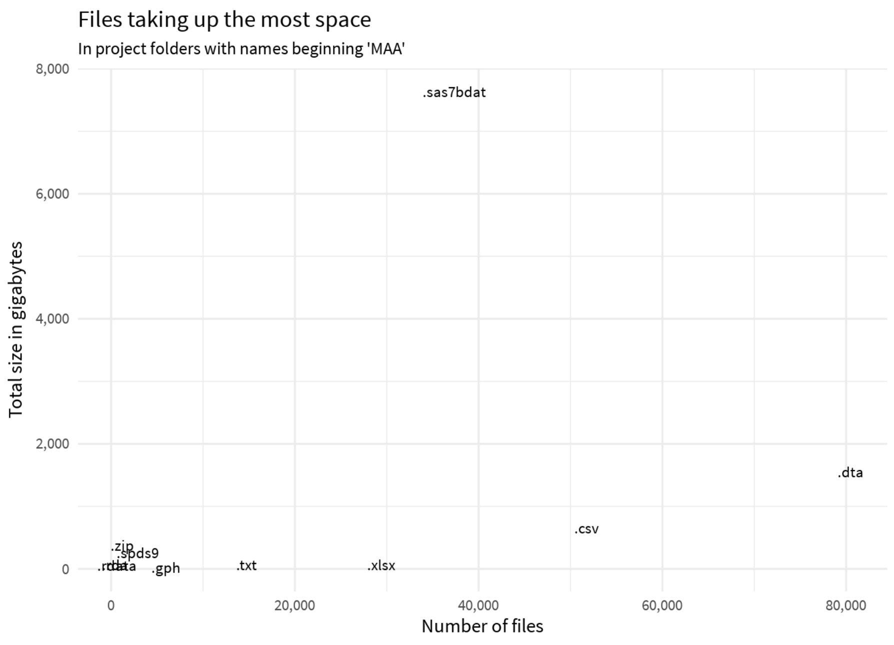
Critically, several projects are far beyond the scale (in size and complexity) where they could be managed without version control. At least two project teams use version control systems outside the Data Lab (releasing code through the output checking process and manually committing it to a version controlled version outside the lab), with considerable cost in terms of coordination and risk.

> *"No, we don't have a SAS style guide. Definitely view ourselves as researchers who code a bit, not as programmers.""*

The standard of code varies. Best practice code, which can be found in many of the research projects folders, is very good indeed and includes all the desirable features set out in the introductory section of this report. However in general, the standard of project work flow and coding falls short of acceptable (never mind good) practice. The root cause is lack of version control software, and without it researchers cannot be blamed for problems such as multiple non-definitive versions of scripts, personalised folder structures, poor use of joint assets, and lack of jointly-developed and maintained functionality.

5.2 Data

Based on analysis of the suffixes of file names, the next chart shows that researchers have saved more than 7 terabytes of data - 15 times the size of the original IDI - from SAS in the form of 40,000 `.sas7bdat` files. There are also 80,000 `.dta` files (Stata's data format), although they take up a much more modest 1.5 terabytes.



To make this less abstract, here are the 20 largest files on the file server:

name	Size (GB)	Number of files with this name
fact_inc_.sas7bdat	83.0	2
master_data_20160802_segments.sas7bdat	39.6	1
master_data_20160725.sas7bdat	39.5	1
master_data_20160616.sas7bdat	39.4	1
master_data_20160621.sas7bdat	39.4	1
master_data_20160802.sas7bdat	39.4	1
combined_data2_2014lag.sas7bdat	36.6	1
scenario314_333pop.sas7bdat	34.2	1
combined_data2.sas7bdat	32.4	5
incpbn.sas7bdat	31.1	1
combined_data2_segment2.sas7bdat	28.7	1

name	Size (GB)	Number of files with this name
combined_data2_segment.sas7bdat	28.0	1
combined_data.sas7bdat	26.5	1
annual_dataset_v8.sas7bdat	25.6	1
pho_status_2008_2015.sas7bdat	25.1	2
IDI_20160224_firm_size_desc_fe.csv	25.1	1
costlong_temp2_314_333.sas7bdat	22.8	2
ir_employee_chars_final.sas7bdat	18.8	1
logisticvar2.sas7bdat	16.7	1
PHARMAC_full_16_3_17.dta	15.6	1

We don't know exactly what is in each but we can get an idea. For example `logisticvar2.sas7bdat` *probably* holds the explanatory variables for a logistic regression (must be a large one; 500 columns of numbers for 4 million people would be about that size). `PHARMAC_full_16_3_17.dta` is probably a copy of the fact table showing individuals purchasing pharmaceuticals (either filtered or very well compressed by SAS, as the original is much bigger than this). And `master_data2016XXXX.sas7bdat` is probably a carefully and expensively constructed dataset, joining and filtering many tables from the IDI, that the analyst wants to store for repeated use.

5.3 Programming languages

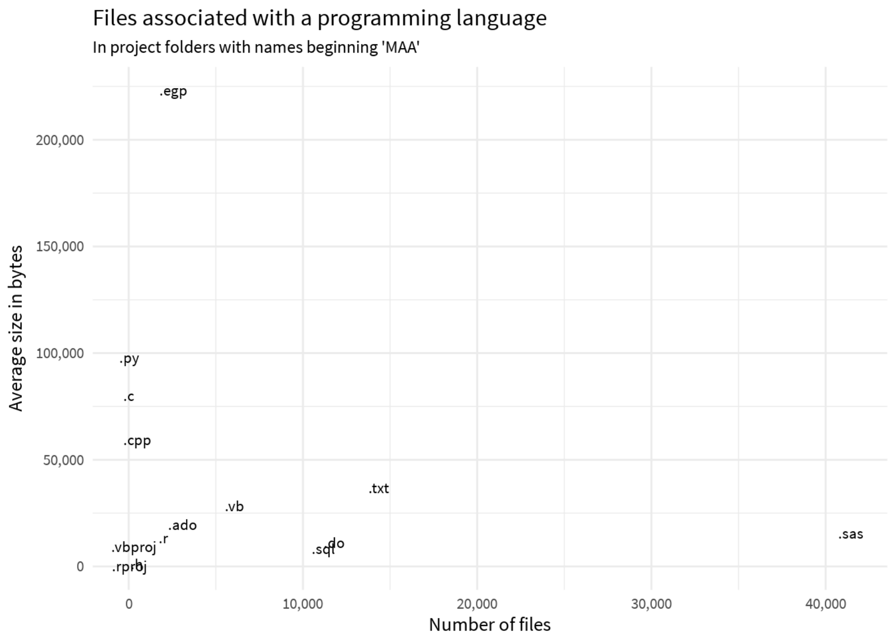
> *"We do all the data pulling and prep in SAS, then present and confidentialise in Excel."*

Of particular interest for our purposes is what sort of programming scripts are being used. The tools available for analysis by regular users in the Data Lab are SAS, SQL Server, Stata, R and Excel. R users can and do (but not very widespread in the Data Lab) also compile C++ programs via the `Rcpp` package, and fit Bayesian statistical models specified with the language Stan.

To a certain extent, we can identify which languages people are programming in by the suffixes of the files they save. There are limitations, because this is just a convention. Some large binary files that are definitely not R programs have been saved by one of the projects with the `.r` suffix; and some SAS, SQL and R scripts have been saved with the `.txt` suffix (rather than `.sas`, `.sql` or `.r` respectively). The `.txt` suffix may or may not be a program (often it could be notes eg `README.txt`, or data), but enough SAS (in particular) programs seem to have been saved with `.txt` suffix to justify including it in this analysis.

SQL is frequently sent to the database server via ODBC (open database connectivity) and the actual code embedded in SAS or R programs (and possibly Stata programs or Excel workbooks, although no cases of these have been found yet) rather than saved in a stand-alone file.

With these limitations, analysis of files' suffixes gives us a starting point for understanding what analytical tools are being used. Here is the frequency with which files associated with these (and a few other tools) appear in researchers' files.



The `.egp` files are SAS Enterprise Guide projects. They are much larger on average than the other programming files because they can include results as well as the researchers' programs.

We have data from the folders of 173 projects (this includes projects that are inactive or complete). Here are the numbers of files normally associated with programming languages of the 15 largest projects:

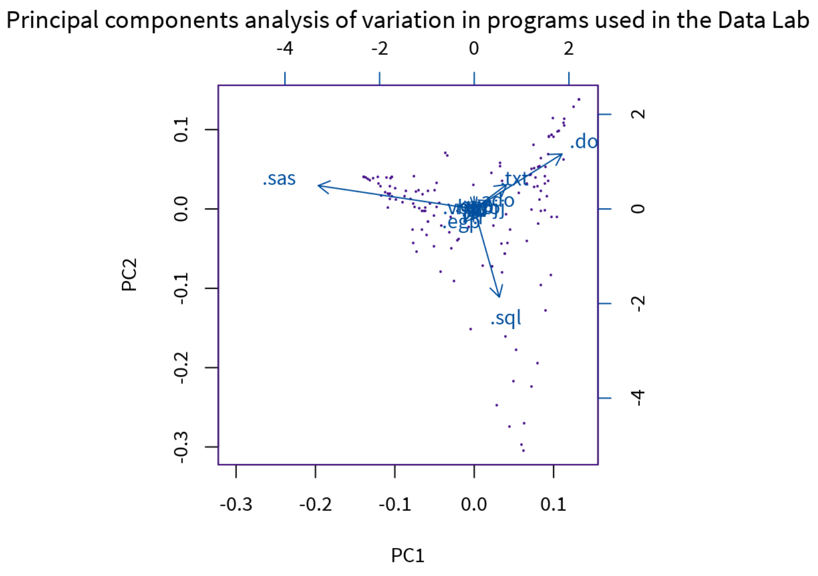
projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P47	2			26	826			571	8	19043	7425	897		
P122	99			375	88			16		4660	87	134		
P105					154		1	440	11	3197	728	174		

projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P125					42					2185	1	20		
P141					73					2024	130	191		
P155		11	483		92	483		3	4	1094	327	1889	6067	284
P51	306			21	12			56		647		60		
P99	116			42	11					638	43	84		
P138	1				13			26	3	607	63	91		
P93	155			231	148			12	1	498	192	589		
P137				263	2		47	10		456		4		
P61					16			24		445	27	32		
P103					3					308		1		
P1					16			34		299	1	30		
P95				2	25					250	2	4		

And here is the simple count of number of projects using each programming suffix:

suffix	Number of projects using
.txt	120
.sas	111
.sql	93
.do	71
.egp	65
.r	46
.ado	37
.rproj	13
.h	2
.py	2
.c	1
.cpp	1
.vb	1
.vbproj	1

To explore a broad characterisation of how different computer languages are used, I extracted the first two principal components from a matrix of the proportion of each project's scripts made up by the various languages. We see from the next figure that research projects can be characterised by those with heavy use of `.sas`, heavy use of `sql`, and heavy use of `.do` (Stata).



Some comments follow on the different languages in use in the Data Lab.

5.3.1 SQL

Structured Query Language or SQL is the only practicable way to extract data from the IDI or LBD. Every project has to start with SQL, even if the analyst sees themselves as a SAS, R or Stata user.

There are less files with the `.sql` suffix than might be expected, given the fundamental challenge of data management in the IDI begins with the relational database. Every research project has to start with SQL, but most of the SQL is inside `.sas` or `.R` (possibly Stata - no examples found) scripts, with data management happening in SAS. Usually the SQL is short snippets for queries written and executived via ODBC with `PROC SQL`. Stata users seem to be more likely than SAS and R users to develop their SQL separately in a `.sql` file (developed in MS SQL Server Management Studio) than to embed it in their main code.

Here are the projects that are the biggest users of `.sql` files so we can see which other programming languages it is associated with:

projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P47	2			26	826			571	8	19043	7425	897		
P105					154		1	440	11	3197	728	174		
P29	617			1652	1						387	1924		
P155		11	483		92	483		3	4	1094	327	1889	6067	284
P93	155			231	148			12	1	498	192	589		
P38	315			506	13			130	1	233	187	682		
P141					73					2024	130	191		
P57					1			2		1	129			
P102	74			414				40		5	112	258		
P114	4			559						1	110	107		

There are 50 projects that *don't* use any `.sql` files. Here is the profile of 10 of them chosen at random:

projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P108				85										
P160								1						
P43					1					3		1		
P9										11		1		
P39								4		12		6		
P128										13				
P71										15				
P66								7		25		7		
P53										137		28		
P137				263	2		47	10		456		4		

Some projects have SQL stored in `.txt` files.

All the research projects start their data journey with SQL, so the absence of `.sql` files indicates more about *how* they use SQL rather than *whether* they use it. Based on the above, on inspection of individual projects, and following up with some researchers, projects with no `.sql` files are embedding their SQL code in other applications' programs. One implication of this is that such researchers are more likely (not inevitably) to be developing SQL in the SAS Enterprise Guide or RStudio development environment, which do not have as much assistance to offer the analyst as Microsoft SQL Server Management Studio which is presumably used by those who save an entire script in SQL format.

5.3.2 SAS

The `.sas` files are SAS programmes and the `.egp` files are SAS Enterprise Guide projects. Users need to open Enterprise Guide to run and develop SAS programmes; but there is no obligation to use its process and workflow managment capabilities.

SAS is very widely used in the Data Lab to create datasets and sometimes for final output directly, which is exported (eg as a `.csv`) or copied and pasted into Excel. Datasets created with SAS end up in one of four states (from apparently the most common to the least common):

- as `.sas7bdat` files in a library in the project network folder, for re-use in an analysis phase with SAS
- as a `.csv` or similar exported file, for re-use in an analysis phase with Stata, Excel or R
- as a table on the SQL Server machine in the "sandpit" database
- non persisting - ie data grooming and analysis stages combined and the data is never saved on disk, only the results of analysis.

SAS users in the Data Lab can be described in three categories:

1. "Pure programmers" who write only `.sas` programmes; in the good practice variant of this category, they use numbered scripts to indicate which order they run in, and a master script that sets up `libnames`, `%includes` source code for macros, and runs the other scripts
2. "Casual Enterprise Guiders" - who save `.egp` files to preserve a snapshot of their work and results at useful moments, but basically work with `.sas` files
3. "Committed Enterprise Guiders" who don't produce `.sas` files at all, but have all their code and processing embedded in the `.egp` file. If these users want to share code they need to export it as a text file.

We can get a crude idea of how many fit into each. Most of the Data Lab researchers are using some amount of dedicated program scripts, with only five projects of pure SAS EG users whereas there are 51 projects of pure programmers:

Type	Number of projects
SAS programs only	51
At least one of each	60
Enterprise Guide projects only	5
No SAS at all	57

Like SQL, SAS is so important in the Data Lab that it is interesting to look at the 27 projects that *don't* use it but do use at least some programming files (note - 30 of the 57 projects that use no SAS at all also don't have any of the files I've defined as "programming-related" at all). Here is a random selection of 10 of those non-SAS but programming projects:

projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P113	10			24							16	33		
P116												2		
P132											18	2		
P145								17						
P82												4		
P106				1							1			
P112				32							19	16		
P19				30							16	5		
P161	10			12							1	5		
P149	17			188							19	878		

We see this sample of non-SAS users mostly rely on Stata (`.do` and `ado` files) in combination with SQL, with one project using R (`.r` and `.rproj`).

During the qualitative phase of this analysis it was noticed that a significant number of SAS programs are saved with the `.txt` suffix. A large number of these programs are copies of templates by the Treasury's Analysis and Outcomes team, which are published in several locations (with the `.txt` suffix). It seems unlikely that people are actually using `.txt` files in SAS directly without either adapting and renaming them as `.sas` files or integrating into an `.egp` project. These files are ignored in the counts of SAS programs above, but are brought back in for the analysis of files with duplicate names in a subsequent section of the report.

5.3.3 Stata

After SQL Server and SAS, the next most used application is Stata. There are slightly over 10,000 Stata scripts - about a quarter the number of SAS programs.

Stata can connect to a database directly via ODBC (http://www.stata.com/meeting/portugal15/abstracts/materials/portugal15_sousa.pdf) but I did not find any examples of this in use in the Data Lab. Unlike for SAS and R, there are no snippets on the "Wiki" explaining how to do this. Instead, data management is apparently typically done either in SAS or MS SQL Server Management Studio with results saved as `.csv` or similar. Secondary data manipulation done in Stata is what leads to all those `.dta` files being saved, recording the state of data during the analysis phase.

There are 54 projects with more `.do` files than `.sas` . Here are the first 15:

projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P29	617			1652	1						387	1924		
P85	75			1524						75		1128		
P90	75			1524						80	1	1130		
P62	66			788						3		238		
P114	4			559						1	110	107		
P102	74			414				40		5	112	258		
P30	22			409						6		356		
P54	40			430						32		290		
P44	44			406	1					30		277		
P38	315			506	13			130	1	233	187	682		
P55	79			226				52		23	58	119		
P135	142			200				1			68	24		
P149	17			188							19	878		
P77	69			153						33	2	13		
P124	209			106				2		5	66	151		

Inspecting some of these projects' folders they seemed to cover quite a range of researcher types, data and questions. I had started with the impression that Stata is disproportionately in use by LBD rather than IDI researchers, but these folders were using a range of both datasets and any difference is not striking. While most LBD researchers use Stata (only a trickle of R and SAS users), I don't think it's true that most Stata users are only in the LBD; the IDI has its share.

5.3.4 R

Around a quarter of projects use R. Most of these use SAS as the main data extraction and managing tool, and R for some specialist statistical modelling or graphics.

To get a brief overview of those projects that use R more than in a niche capacity, here is the profile of the 16 projects that use more `.r` than `.sas` files

projcode_conf	.ado	.c	.cpp	.do	.egp	.h	.py	.r	.rproj	.sas	.sql	.txt	.vb	.vbproj
P70					3			149	2	6	20	2		
P72				51	13			143	5	106	6	21		
P102	74			414				40		5	112	258		
P55	79			226				52		23	58	119		
P143				51				27		5	1	2		
P136					25			19		1	82	1		
P145								17						
P7								38		29		6		
P35	3			6				7				2		
P133	17			34				7		1	2	1003		
P115								11		6	6	5		
P4								2	1			1		
P57					1			2		1	129			
P135	142			200				1			68	24		
P160								1						
P80								1				8		

Very few projects use RStudio projects; and those RStudio projects that were inspected closer were not making the most of their full capabilities. For example, working directories tended to be specified in full, absolute paths rather than taking advantage of the portability a project provides.

Very little data is saved in R's native `.rdata` data format.

While preparing this report and other analysis associated with my project, I built familiarity with the IDI and LBD, and reflected on my own workflow (other than when I was simply adapting and re-running SAS code written by others). I found myself doing as much work as possible in MS SQL Server Management Studio, writing SQL where I could make the most of its support (such as syntax completion, table exploration, canned queries, standard reports, and query execution plans). I used R only for analysis of completed data sets and the production of final products like graphics and presentation tables; and for joining together repeat tasks. For example, I had an R script that sequentially called 13 different SQL programs to measure database performance each day. So typically, I had both SQL Server Management Studio and RStudio open. I worked with a folder system for each of my sub-projects, and an RStudio project for each folder system which meant that within R I could use relative file paths and make use of functionality such as "find all" to find all occurrences of a string in the sub-project. If version control were available I would have had one repository per sub-project too. This (apart from the lack of version control) was an effective (but by no means the only) way to manage workflow and code development, and looks similar to that used by many LBD users (except with Stata dominating rather than R).

5.3.4.1 Other

There are 23 programs written in Python with up to six (but mostly two) copies of each program. All but one of these Python programs were associated with a single large simulation project.

There are 28 programs written in Visual Basic, most of them with hundreds of copies each as part of a workaround of there being no version control in the Data Lab. These are all in one project which is atypical in its toolset and was off limits for closer investigation as it was marked as sensitive.

5.4 Performance

5.4.1 The symptoms

Some researchers complain of poor system performance, although it should be noted that this is generally a secondary concern and *not* the single area most of them would prioritise for improvement.

> *"about the Data Lab environment running slowly over the past few weeks… At its worst it: - can be very slow trying to navigate wprdfs08 folders in Windows Explorer - can take a couple of minutes to open a small file on wprdfs08 (e.g. XL) - can be incredibly slow to edit / save a file on wprdsf08 (e.g. Word document) – sometimes ending with the user being "kicked out" of / disconnected unexpectedly from the Data Lab - slows down our programs (whether R or SAS) when they're reading data from / writing data to wrpdsf08"*

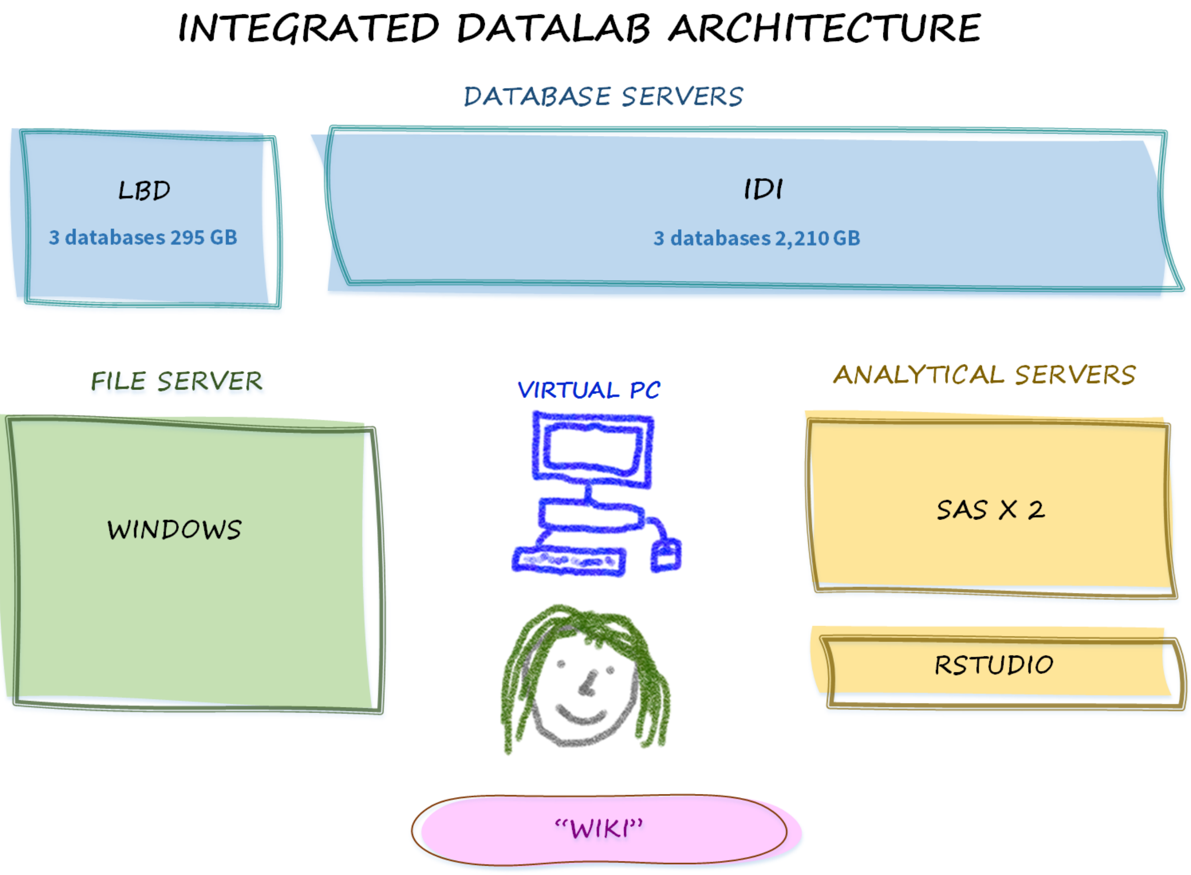
System-wise, speed of running an analytical project faces three potential system bottlenecks:

1. moving data around between servers on the Data Lab network and writing and reading it to and from hard disk
2. computationally intensive statistical methods such as bootstrap, fitting models with Markov Chain Monte Carlo methods, or simulations
3. data manipulation, re-coding and re-shaping

From observing how analysis works in the Data Lab, there are issues and opportunities in each of these three areas. I will address them in the sequence above, which is ordered by significance of the impact on performance.

5.4.2 Moving data around

Moving data around on the network and writing it to disk is a major cause of slow performance. In running researchers' code, it was common to observe examples such as processor time of 2 minutes and elapsed time of 30 minutes, with nearly all the time spent writing a large file to the file server `wprdfs08`. The problems can be visualised with the help of an even further simplified picture of the Data Lab archtiecture:



> *"…opening / reading / writing files (data, programs, documents etc.) on wprdfs08 is the root cause…"*

Every time data moves from one server to another there is a delay as it travels through the "pipe" that is the Data Lab network. Every time it is written to disk there is even more of a delay.

> *"I would say since the earthquake, having the data on a separate filer server has been an issue for two reasons - the 'network' we have is not doing all that well, and volumes of data being sucked off the SQL servers has increased." (Stats NZ Digital Business Services staff member)*

Researchers have saved nearly 8 terabytes of data in SAS format on the file servers (more than 15 times the size of the entire original IDI), effectively treating them as a database that it is not designed to be. This has reached the point where it compromises the file server's main function; even small documents can be slow to open, or the system can break down. Upgrading the file server (as is being done while this report is finalised) will help but not address the underlying problem. If researchers could be encouraged to perform as much data reshaping as possible in SQL rather than SAS, and to save data in the Sandpit on the database server rather than in SAS format on the file server, there could be significant gains in overall performance. At the moment, the settings and guidance for the Data Lab, as well as legacy code and culture and researcher familiarity with SAS rather than SQL, do not encourage this.

The bottom line is that problems in this space can be mitigated by moving the analysis closer to the data (ie onto the database, written in SQL and executed on the `wprdsql36` or `wprdsql31` server); and in particular by avoiding unnecessary movement of large datasets over the network and writing to disk. This is one area where practice by LBD researchers is closer to optimal than that of IDI researchers; LBD research is typified by most of the heavy lifting with regards to data management being written in SQL scripts executed on the database server.

5.4.3 Computationally intensive

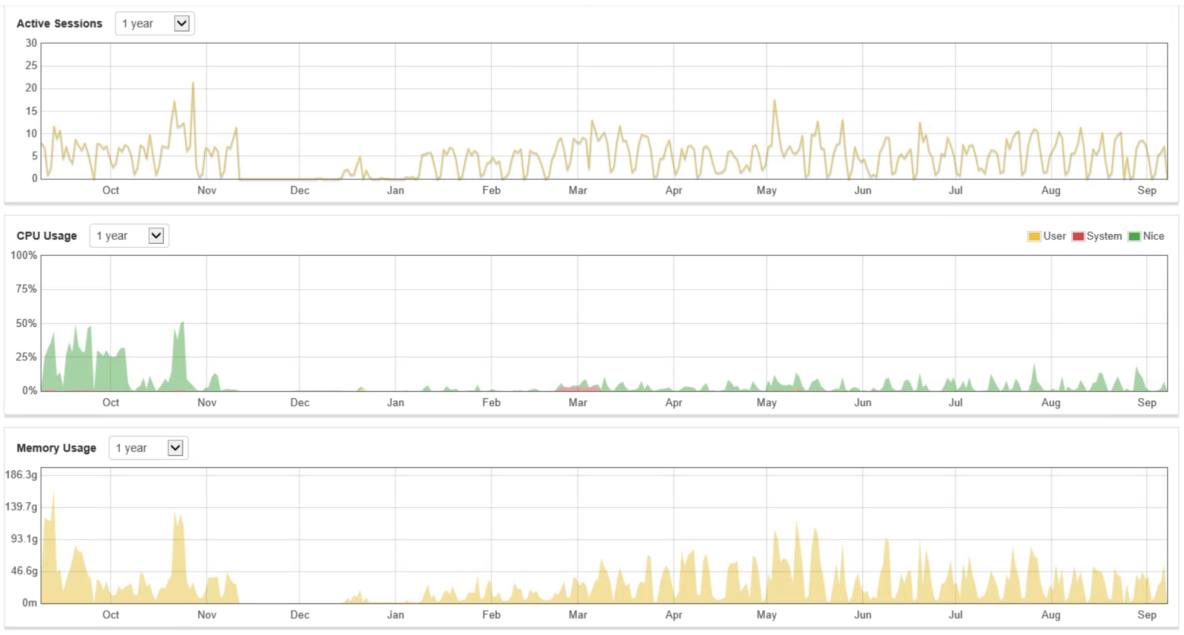
Some researchers need highly computationally intensive operations for their analysis and report that it takes a long time to run. While I have not made any attempt to quantify the problem, from inspecting code and talking to researchers the problematic operations are (most serious first):

- Fitting Bayesian models with Markov Chain Monte Carlo (eg via R and Stan) when there are many parameters and large datasets
- Complex individual-level simulations (eg for quarterly observations for all current and as-yet-unborn children for the next 20 years)
- Bootstrapping (ie resampling with replacement many times and fitting a model to the resampled dataset)
- Fitting individual models to large data

These methods are mostly done with one of R, Stan, SAS and Stata. Some of the biggest tasks are in R, including microsimulations by large government research programmes, and MCMC fitting of large models (eg thousands of parameters, millions of data points) by the Stats NZ Census Transformation research.

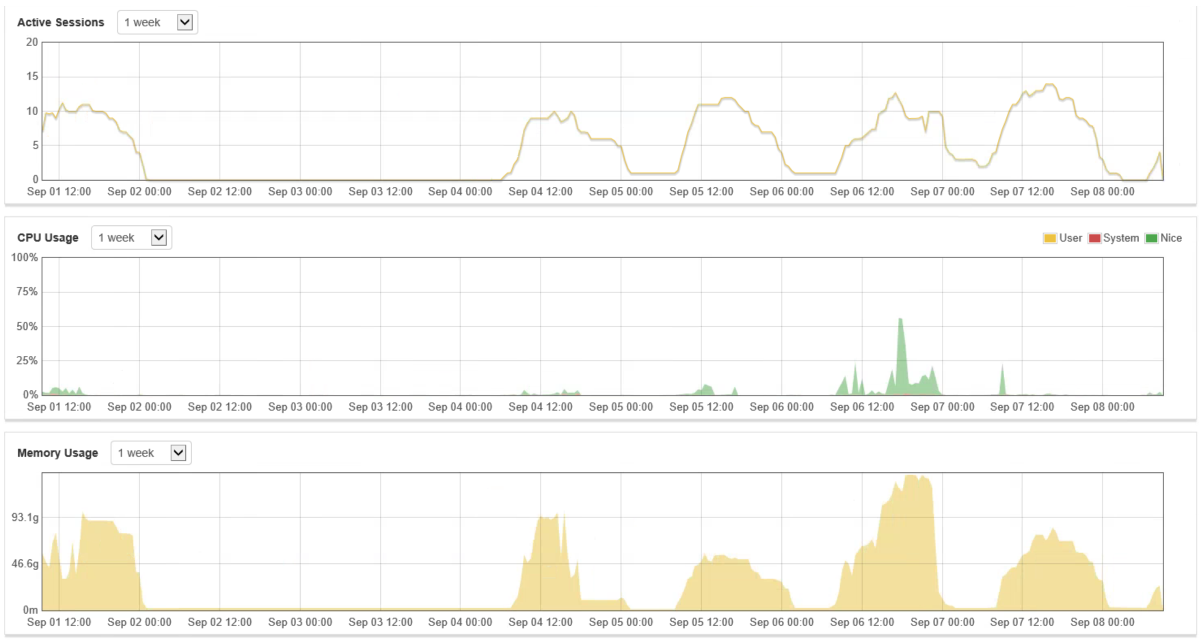
5.4.3.1 RStudio

I wasn't able to obtain good information on servers' capacity and use for these purposes other than for RStudio. Capacity/use statistics for the `rstudio03` server over past year are shown below:



The `rstudio03` server is a substantial machine with 36 CPUs and 138GB of RAM. Memory is often the limitation for R when working with large data, as the standard programming methods mean entire datasets are held in RAM at once. This can lead to very fast performance when appropriate methods are used but creates an upper limit on some operations with larger data. We see that on a few occasions shortly before the 2016 earthquake the server approached its maximum use of memory over a period of several days, but in general this is not a sustained issue.

However, when we look at a more fine grained picture of the last week we see that in a typical day there are spikes when the server gets towards its full capacity. For example, the spike in memory usage seen in the afternoon of 6 September took the server briefly to its maximum capacity and is visible on the weekly chart, but does not show when the log is monitored only at the annual level.



Users reported that they had to modify behaviour to allow for limitations on the single R server in the Data Lab:

- one group of researchers split their tasks into separate jobs and recombine the results when complete (this is in fact the standard approach to dealing with large data problems, but the setup in the Data Lab does not have some of the tools used outside to make this easier)
- other researchers reported closely monitoring system resources on the R server and limiting the numbers of CPUs they used in parallel processing jobs to allow for other users (as the instructions on use of `rstudio03` set out).

5.4.3.2 SAS

SAS users also reported some lengthy (ie duration in hours) statistical tasks.

Stats NZ has proposals to implement SAS Grid. In a SAS Grid environment (http://support.sas.com/software/products/gridmgr/index.html#s1=1), workloads are distributed across a grid of computers. This could mean a collection of SAS servers, or co-location with a Hadoop cluster. This workload distribution enables workload balancing, accelerated processing, and job scheduling. Depending on how this is done, it could definitely speed up those computationally intensive applications that are run on SAS (which is by no means all of them - SAS is much less dominant in computationally intensive analysis in the Data Lab than it is in the management of data to the analysis stage).

5.4.3.3 What is to be done?

There is no magic fix to the problems experienced by both R and SAS users in this space. From inspecting researchers' code, there are in some cases improvements in efficiency to make, but some code is already optimised and simply takes a long time to run with existing resources.

A long term fix in this space might involve a significant shift in tools and infrastructure. Researchers would need:

- permissions and tools to (on a user pays basis) scale up infrastructure to a large number of processors and RAM - the equivalent of spinning up a new cluster of servers on a cloud based service like Amazon Web Services or Azure
- ready access to tools that embrace the "divide and recombine" approach to analysis with large data, like DeltaRho (http://deltarho.org/) which uses R in combination with Hadoop;
- tools which can fit complex models efficiently on a cluster of servers, like H2O (https://www.h2o.ai)
- skills to make the most of these tools.

These may not be realistic hopes for the near future.

Medium term increases in capability might include installation of H2O on the RStudio server and implementation of SAS Grid to enable parallel processing in SAS (http://blogs.sas.com/content/sgf/2013/10/02/parallel-processing-in-sas-when-and-how-to-use-it/) to match what is already available for R users.

In the meantime, improvements that are within the power of today's researchers could include:

- more use could be made of R's parallel processing capabilities (https://cran.r-project.org/web/views/HighPerformanceComputing.html), which are considerable and need only minor coding adjustments to work (bearing in mind that no single R job should use more than three quarters of the processors available on the `rstudio03` server).
- there are sometimes big pay-offs from refactoring Stan code (used for Bayesian models) so random variables are vectorised and parameters are standardised. Stan also needs to be explicitly told to use parallel computing eg via `options(mc.cores = 4)` and some Stan code I observed was not doing this.
- some R packages are much better at dealing with larger data faster than others. For example, the `ranger` R package will fit random forests to data where the `randomForest` (used in at least one IDI project looked at) will slow down to impractical levels or fail.
- For some problems, R users could also make more use of `Rcpp` to write code in C++, which compiles successfully in the Data Lab. One example was found of a Stats NZ research using this capability and there is scope for more.
- the lightning fast xgboost machine learning algorithm already works well on `rstudio03` and more researchers could make use of it (at least one project was found already doing so)
- fitting models to smaller sampled datasets and using statistical properties for inference to the full population.

5.4.4 Database performance

While data reshaping and coding operations are not the main cause of any performance problems, there is definitely room for improvement. The performance of the database servers is slower than might be expected, sufficient to be a frustration during iterative development of analytical queries.

During the research for this report I timed the daily performance of 13 simple SQL queries. Queries that were just joining two tables together (and in some cases not even that) and aggregating by obvious groups would consistently take three to fifteen minutes to run. 10+ minutes for a query with two simple joins on the database's most obvious primary key (`snz_uid`) is poor performance, even on tables with tens of millions of rows. SQL Server is routinely used for much larger databases than the IDI with much better performance.

Neither the IDI nor the LBD have been designed or tuned post-design for performance.

In terms of design, the wide fact tables for surveys and a few of the administrative data sources may be convenient for how people are used to thinking of the data, but aside from the design stability problems mentioned earlier, they probably reduce performance relative to what would be the case with more traditional (in database terms) longer, thinner, more normalised tables.

In terms of tuning, many obvious indexes that would almost certainly speed up queries materially have not been added (some are being added now, as I have engaged with the Integrated Data team and with the database administrators in the course of this research). This has a major impact on database performance. Annex D provides a small case study where run time of simple queries on modest sized tables (4.5 million rows) could be reduced from 15-30 seconds to 0 seconds by adding an index on the right variable in addition to the primary key.

One table that was meant to have a primary key was found not to (this was quickly fixed by the Integrated Data team). The fact that this was even possible indicates a flaw in the extract-transform-load processes which should be considered when it comes to the IDI 2 redesign. Primary keys and indexes should be added programmatically and automatically when data is loaded.

Tables in the LBD at least have primary keys. Most tables in the IDI do not have even that, but are unindexed "heaps". Some indexes are being added, but more work is needed to add the right ones that will help common queries, particularly if more data reshaping work is done in SQL as I recommend to take some burden off the file server.

In preparing this report, I several times inspected the `wprdsql36` server's reports to see which queries were taking much time. The vast majority of queries take less than 10 minutes to run. The slowest query I observed took about two hours to run. It was drawing on views in the researcher's sandpit area that draw on the large pharmaceuticals table (largest in the database) and is reproduced below:

```sql
select
  sial.*,
  inp.as_at_date,
  dateadd(yy, -2 * 1, inp.as_at_date ) as profile_start_date,
  dateadd(yy, 0 * 1, inp.as_at_date )  as forecast_end_date

from      [IDI_Sandpit].[DL-MAAXXXX-XX].SIAL_MOH_pharm_events    as sial
inner join [IDI_Sandpit].[DL-MAAXXXX-XX].si_2013_cohort_char_ext as inp
on ( sial.snz_uid = inp.snz_uid and
    sial.start_date <= dateadd(yy, 0 * 1, inp.as_at_date ) and
    sial.end_date >= dateadd(yy, -2 * 1, inp.as_at_date ) )
```

The right indexes on the tables on `IDI_Clean` that these views are drawing on would help, and even better would be if the views could be converted to being indexed views. More discussion will follow in the Opportunities section later.

6 The research community

> *"IDI users are very very diverse"*

6.1 Documentation and guidelines

The bedrock of an analytical community is whatever formal documentation exists, whether this is generated by a central body or evolves organically from the community itself. Both the LBD and the IDI have a lot of documentation and metadata available but it is not complete, and not optimally organised, so the impact can be overwhelming.

> *"There are so many documents all over the place – data dictionaries, example code, wikis, discussion forums – not possible to take it all in, which is why you need someone experienced … to show you where everything is."*

The LBD has an excellent researcher-oriented introduction, Fabling and Sanderson's *Rough Guide to the Longitudinal Business Database* (http://www.treasury.govt.nz/publications/research-policy/ap/2016/16-02).

The IDI does not have an effective manual or users' guide oriented to researchers. Several projects have made their own versions of an induction manual, with starter information such as the location of servers, which datasets are commonly used in "our" project, and some example SQL or SAS code to get new project members started. None of these documents to my knowledge claim to be particularly satisfactory. Stats NZ commenced work on a more comprehensive guide but it is currently on ice, pending prioritisation as part of the IDI 2.0 project.

The documentation and guidelines that *does* exist are in seven broad categories, set out below.

6.1.1 Stats NZ website "Resources for IDI users"

The Stats NZ website is the starting point for potential researchers and is available to everyone. Relevant pieces of guidance include:

- invaluable data dictionaries (http://www.stats.govt.nz/browse_for_stats/snapshots-of-nz/integrated-data-infrastructure/idi-data.aspx) for most of the core datasets, but not the ad hoc datasets
- presentations and papers on the linking methodology. These are useful and important, but are written from the point of view of methodologists doing the linking rather than practitioner researchers using the results.
- the microdata output guide for confidentialisation, etc
- some SAS/SQL code snippets from Treasury and from Virtual Health Information Network (VHIN) as downloadable files.
- links to (or at least how to contact)
  - Meetadata
  - the Social Investment Measurement Map (SIMM) on the Social Investment Agency (SIA) website
  - SIA's GitHub page, which has code to create the Social Investment Analytical Layer
  - Some generic code tutorials for SAS, SQL and R

6.1.2 Other websites

The best general introduction to the IDI for researchers is the Virtual Health Information Network's "Guides" (https://vhin.co.nz/guides/getting-started-using-the-idi-in-the-statistics-new-zealand-Data%20Lab/). These have practical information on details such as how to connect, what language to learn (SQL as top priority), what the spine is and how to know if a cohort of people known by `snz_uid` is on it, etc.

SIA have the "Social Investment Measurement Map" guide (SIMM) (https://www.siu.govt.nz/tools-and-guides/measurement-map/simm-guide) which is a convenient (searchable, combined in a single table, and sortable) cut-down version of some of the metadata on some of the IDI datasets. It aims to help potential researchers scope possible analysis. However, it has only partial coverage, and not as much information as is in the full metadata (which is in inconvenient PDFs on the Stats NZ website).

Various other sites (eg Treasury) have some useful articles on methodology and pieces of guidance.

6.1.3 Meetadata

Meetadata is a Sharepoint site available to researchers via application and RealMe log on. Like the other sites mentioned so far, it can't be accessed from the Data Lab when researchers are actually undertaking work. Meetadata:

- looks most useful for announcements and publishing slides from user forum sessions
- has the same code snippets as the website, with the same issues (has to be downloaded rather than viewed in browser)
- some under-used discussion boards, split into too many categories, difficult to navigate, clunky to access attachments
- some useful documents hidden away in non-obvious places eg in the Health discussion board are some "getting started" documents introducing SQL and the IDI

Meetadata is not popular with researchers spoken to for this report, because of its interface (perceived as clunky), the need for RealMe logon, and because it is not available in the Data Lab.

6.1.4 The Data Lab "Wiki"

All researchers have access to a Sharepoint "Wiki" when they are in the Data Lab (but not otherwise). It:

- is by far the most complete "all in one spot" resource on guidelines for access, general info about the IDI and about the LBD, ad hoc papers on data quality issues, meta data, privacy, confidentiality and data release, published research, coding tips (eg "converting date variables in SAS"), survey questionaires, etc
- has code snippets which look runnable, and in many cases do in fact run if the right environment is set up
- is not really a Wiki – general users can't edit (other than the discussion board), but have to submit items to Stats NZ to be used
- shares code via a "discussion board", which is not well categorised, threaded, or good at handling versions, but is searchable
- is difficult to navigate
- is on a platform that is not suited for sharing code or for building knowledge through Q&A, compared to modern tools and expectations formed with GitHub (https://github.com) or Stack Overflow (https://stackoverflow.com/). For example, code can only be attached, not viewed inline with numbered lines and syntax highlighting as per modern browser experiences. Even as a commenting platform the Wiki leaves much to be desired - for example, formatting options for things as simple as paragraph breaks are not obvious and in many cases researchers have clearly not found them.

The Data Lab Wiki has the physical model of tables in the IDI and their columns but it is out of date (lists some views which are not there) and not particularly tractable - 57 pages long. Better would be to release a CSV with the tables and columns as an additional, searchable tool.

The version of Sharepoint in the Data Lab is quite old, which is partly the cause of some of the difficulties with its usability.

6.1.5 Github

Stats NZ and SIA both have GitHub accounts and have published code on it:

- SIA
    - SIA have published code standards for SAS, SQL and R and their published code mostly meets them; recently theyeven published a[] constructive self-criticism](https://nz-social-investment-agency.github.io/sia_analytical_processes/output/coding_style_critique.html (https://nz-social-investment-agency.github.io/sia_analytical_processes/output/coding_style_critique.html)) of some of their own earlier code against those standards, partly to encourage others to also seek to raise coding standards
    - SIA's code is runnable by others, but it has to be downloaded as a zip from the internet and dropped into a researcher's folder in the Data Lab to be used
- Stats NZ
    - Code on the Stats NZ page is the same as on the website and Meetadata (Treasury's childhood factors, and VHIN's something else), with the same problems; but the additional issue that it is posted as zip files rather than text, hence still not viewable in the browser

6.1.6 Personal contact

In the absence of written documentation, researchers rely on personal contact:

- Most researchers have not published their code but are sometimes prepared to share it if personally approached and they have the opportunity to explain it.
- Researchers with contacts in government agencies can (sometimes) contact knowledgeable people in involved in the data generation processes to resolve questions about metadata, interpretation, etc.

> *"Some of the new money should be spent on an introductory course, rather than throwing people in and wasting tens of thousands of tax payer dollars on floundering around and building knowledge of the metadata. A 2 day hands on course on how to use and interpet the the metadata, put together a simple table, avoid tripping up – would pay for itself easily. Yes a manual would be helpful too to remind people, but the course would be the main thing. Currently it's all down to personal connections to get to understand those things and it's incredibly inefficient."*

6.1.7 Unpublished Stats NZ resources

Unpublished or partially available documents held at Stats NZ include:

- eight user guides by various experienced researchers that Stats NZ has accumulated. These all have useful parts but form a mixed bag of administrative ("how to add a new researcher"), conceptual issues, and generic code ("this is what SQL is").
- various feedback that has been collected on the "new researcher" experience
- an incomplete guidance document for new IDI researchers, drawing on the above.

> *"Code sharing is greatly complicated by so many different versions going around, and different locations hosting the shared code (meetadata, website, Wiki, github, personal communications being dropped in, etc)."*

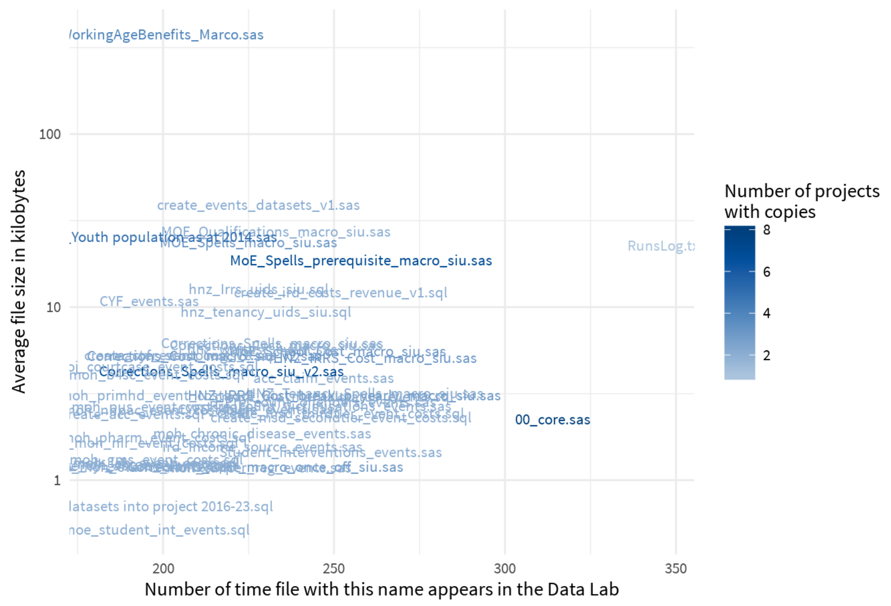
> *"We see code sharing as a head start to adapt and build on, not as something you run out of the box. Wouldn't just run someone else's code as much as use it as a starting point, carefully check every line. Certainly not happy using a dataset created by someone else."*

6.2 Seeking shared code

Another key element of a successful analytical community is how it shares code, and even more so, engages in joint development and evolution of a community code base. To start understanding how this worked I looked for the genealogy and DNA in computer programs with shared nuggets of code.

As a starting point I tried identifying files that appeared multiple times in researchers' folders. When I confined the analysis to file suffixes associated with programming languages but excluding Visual Basic, I identified a number of files that appear more than 100 times in the researchers' directories:

Coding files with identical names appearing multiple times



For example, `00_core.sas` is in eight projects. Rather than existing as many exact copies, variants of it are being used as a controlling SAS script to run other SAS scripts. The original version of `00_core.sas` comes from the Treasury Analytics and Insights team and is shared in multiple locations, so it is not surprising that many projects have copied and adapted it for their own file paths and libraries. So it's more a naming convention (and a good one) than genuine duplication. One project has 284 files called `00_core.sas` as a work around for having no version control software (one of the large government research programmes). Nearly every copy is in a different back up version of a major analytical product has its own folder system with a `./sql/` folder and a `./sasprogs/` folder.

Some of the SAS programs that obviously define macros or populations are of interest, for example:

- `WorkingAgeBenefits_Marco.sas`
- `MoE_Spells_prerequisite_macro_siu.sas`
- `CYF_events.sas`

These are indicators of people re-using code, either formally as a macro with parameters, or adapting past template code to create similar populations to those done previously.

Here are the programs written in SAS, Stata and R that are copied by at least five projects, excluding those associated with building the Social Investment Analytical Layer as a special case (it has been imported into the Data Lab by around ten projects, and steps are already under way to include it in a central database accessible to all researchers). Nearly all these are SAS programs. The first four are copies of the random rounding macros for confidentialisation of output. `libnames.sas` and `FORMATS_new.sas` are used as part of the session set up and for matching data to metadata. The table below is dynamic and can be extended to show all those files:

Show 15 ▾ **entries** **Search:**

	name	freq	likely_mean_size	number_projects
1	RR3.sas	124	7182	55
2	GRR1000.sas	48	8465	46
3	GRR10.sas	48	8089	46
4	GRR_for_LBD.sas	49	7048	45
5	Pent_RF.sql	47	5455	45
6	R5_for_unweighted_counts.sas	46	4985	44
7	Std_macros.txt	62	12114	12
8	02_Health_indicators_MentalHealth.sas	52	26426	10
9	CYF_client_events.sql	27	1990	10
10	FORMATS_new.sas	82	8649	10
11	macro_AdultSpellsonYouthWorkingAgeBenefits.sas	23	390075	10
12	MOE_macro_new.sas	76	31055	10
13	CUST_macro_new.sas	74	4696	9
14	CYF_macro_new.sas	80	35885	9

	name	freq	likely_mean_size	number_projects
15	CYF_rules_macros.txt	37	11083	9

Showing 1 to 15 of 75 entries Previous 1 2 3 4 5 Next

The high number of occurances of some scripts compared to the number of projects they are in is again a symptom of a version control work-around. In the absence of version control software, researchers are making copies of a complete folder system and date stamping it; this leads to proliferation of copies of collateral (such as template or example scripts they have copied from the Data Lab Wiki).

The files listed in the table above should be seen as a starting point for primary candidates for agreeing a community-wide definition and including in a standard centralised library of functionality. Perhaps of more general functionality available to R and Stata users as well as SAS users.

> *"More complex constructed variables like 'highest level of education attained' are very very difficult to construct – eg knowing which categories to collapse into which others – without the benefit of expert guidance from people who use the data and know the 'normal' way to do it. Very dependent on finding the right individual."*

Many but not all of the files listed above come originally from Treasury Analytics and Insights team. This exercise also draws attention to the two main sources of copied files, and in effect the existence of two ecosystems - one based on Treasury and one on the Social Investment Agency. A fair number of the scripts shared in the "Wiki" (see below) are dependent on the standard Treasury macros and libraries being available; and the the SIA's code for analysis is dependent on macros in the "Data Foundation", which in turn is dependent on the SIAL being created. It is quite possible to work with both eco systems, and there are plenty of projects that use the SIAL as well as Treasury-derived macros; but it is worth noting the complexity of maintaining these two different systems across a rambling folder system.

The lack of version control software in the Data Lab exacerbates this problem greatly - a piece of shared code may depend on the standard Treasury macros, but which version of them? And how is a user to efficiently ensure they have the latest version, without having to change references in downstream programs?

This lead me to an examination of the main media for sharing LBD and IDI code. In the next sections I examine and re-run code published (only visible to other researchers) on the Data Lab Wiki, the Social Investment Agency's GitHub page and elsewhere.

6.3 Code sharing and shared code

6.3.1 The Data Lab "Wiki"

There are 141 postings on the "Discussion Board" in the Data Lab Wiki, which is the easiest way of sharing IDI and LBD code for actual use. All other ways involve clearing the code through output checking on the way out, and asking Stats NZ to drop them back in to the target folder on the way in. The Discussion Board is in Sharepoint and, as mentioned previously, is ill-suited to sharing and viewing code.

The most prolific publisher of code in the Data Lab by far is Treasury, with 94 of the posts; MSD is the only other organisation in double figures (11). Nearly all the posted code is in SAS. Most of the postings are of snippets of code for useful, repeat tasks such as creating indicators based on the "before school" checks. All but three of the postings relate specifically to the IDI; of the others, two are general (eg guidance on "explicit pass through" SAS code to force SQL to be executed on the database server) and only one relates to the LBD.

A few of the postings are for complete end to end analyses, such as the "Treasury Infographics 2017 project", four Virtual Health Information Network (VHIN) programs that were deliberately commissioned as exemplar analytical projects, and the "Stats NZ Otago youth NEET pilot partnership project". This latter is notable in being one of the few pure SQL projects (eleven files) posted.

The code published in the Data Lab "Wiki" is not available outside the Data Lab environment, which is a shame because it could be useful for training purposes and helping potential researchers assess what it takes to do analysis with the IDI. There is no in-principle reason stopping this code from being published; it just needs to be checked that no data is included (none was in any examples I looked at) and a process in place to publish it outside the Data Lab (eg for it to be ported to the under-used Stats NZ IDI repository on GitHub (https://github.com/StatisticsNZ/IDI)). However, the greater availability might unfortunately be a chilling factor for researchers already averse to exposing their code to scrutiny, so positive efforts would be needed to encourage and support analysts in making this transition.

The full list of postings is interesting enough to reproduce here (the table below is sortable and searchable but only shows ten postings at a time):

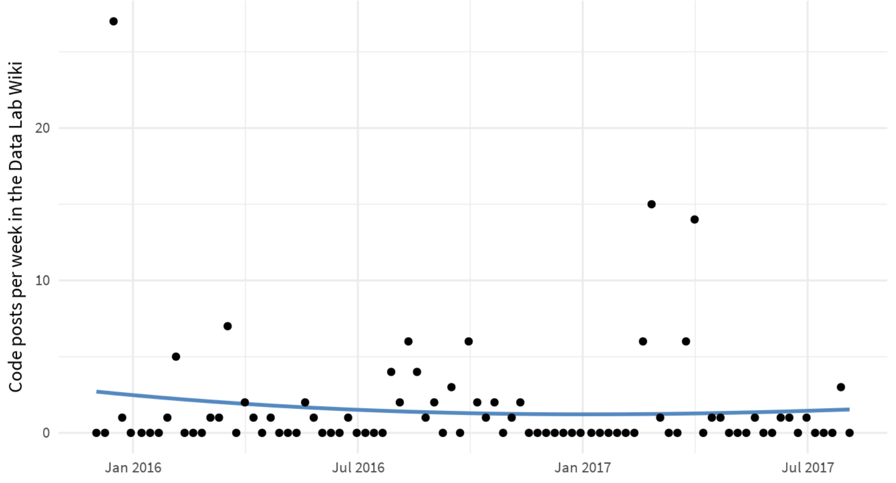
Show 10 ▾ **entries** **Search:**

	Subject	First posted	Relates to	Organisation
1	MoH Parent Child Relationship	2017-07-28	IDI	Treasury
2	Code files for Stats NZ Otago youth NEET pilot partnership -project	2017-07-27	IDI	Stats NZ
3	SOC Priority Population for MSD beneficiaries with young children	2017-07-26	IDI	MSD
4	Code for linkage analysis	2017-06-27	IDI	Treasury
5	MSD Mental health cohort	2017-06-13	IDI	MSD
6	NCEA student performance measure	2017-06-06	IDI	NZ Initiative
7	MMSD benefit (first and second tier) extracts of spells and snapshots for either health condition or mental health condition clients	2017-05-17	IDI	MSD
8	Maori enterprises in LBD - extract and use with AES code	2017-04-18	LBD	Stats NZ
9	TEC Outcomes from Tertiary Education	2017-04-11	IDI	

	Subject	First posted	Relates to	Organisation
10	school leavers code on qualification	2017-03-31	IDI	

Showing 1 to 10 of 138 entries Previous 1 2 3 4 5 … 14 Next

There is no particular trend in posting code over time, other than the obvious gap in the post-earthquake period in which the Data Lab was unavailable.



6.3.2 Mini case studies from the Wiki

The code on the Data Lab Wiki is generally of good quality in terms of style, efficiency and presentation. Posted programs cover a range of tasks. An excellent introduction to the IDI would be to work systematically through as many of the posted code extracts as possible, try to get them to work, and reason through what they are doing and why. Even better would be to translate the SAS code into SQL, refactor them, and streamline for performance.

The programs posted in the Wiki rarely run "out of the box" and at a minimum all except the most portable need some editing of file paths, library names, and schema names. Compared to the functionality available in the outside world (eg the ability to easily install and load up an R package or a Stata library, or to clone a repository of code or a simple Gist from GitHub) the experience is very unsatisfactory. Some small case studies follow.

6.3.2.1 MoJ Reoffending R3.sas

Posted March 2016. Code had been edited in 2015 to refer to the classifications in `IDI_Metadata` and hence commendably make the code portable for non-Justice users. Purpose is to calculate re-offending rates to two years after conviction.

- Successfully creates table with MoJ names.
- Cannot find `metadata.moj_asoc_code.DATA`. Looking at the `IDI_Metadata` there is an ominously named table `moj_asoc_code_till_20170720`, which is a classification table for ANZSOC (ie offence group, division, subdivision, and short verion of division.). Further inspection shows all the IDI_Metadata tables for MoJ have two versions, one labelled _till_20170720, one not.
- Correcting the names of tables from the `metadata` libname to make them refer to the old tables comes up with the new error that `MOJ_CHARGE_OUTCOME_TYPE_CODE_TILL_20170720` is too long for SAS to handle.

The above is a nice illustration of what must be day to day frustrations for researchers, when database designs (ie table names) are subject to change, and people are working across applications (SQL Server, which allows names longer than 32 characters, and SAS which doesn't). Note that this problem happens even when pointing the code at an old datestamped version of the IDI - because IDI_Metadata is not date stamped, but table structures changed as classifications changed.

6.3.2.2 reOffDeathMig_macro.sas

Related to the reoffending code above and posted the same time (March 2016), but abstracted into a parameterisable macro in the interests of having a stable piece of code that does the work, and a parameter in another flow to call it.

Same problem as above of the changing offending classification in `IDI_Metadata` breaking the code.

6.3.2.3 Creating cohort population

Posted February 2016, this script identifies a basic list of people born within the cohort year and on the spine and where there is no evidence from border movements that they were away from NZ for X years with a defined period. Uses several sources to find other active identities - tax, health and MSD. Is parameterisable at the top of the script, but not abstracted into a macro. This program was originally developed for a particular piece of research and was released for other researchers to adapt.

In principle this should run out of the box, once amendments made for location of folders and database versions. I had a problem in that I didn't have access to the `IDI_Clean_20150513.moe_clean.student_enrol` table although I do have it in later years (good example of an odd quirky frustration for researchers); but it ran past that point when I repointed it to `IDI_Clean_20170420`. However, in the end I couldn't complete it because I didn't have access to the `moh_Health_tracker_cost_201503` and `moh_Health_tracker_pop_201503` tables in the `IDI_Sandpit.clean_read_MOH_Health_tracker` - although I *should* have that access according to Stats NZ policies, because I have access to the `moh_clean` schema on `IDI_clean`.

This provides a nice example of reproducible data creation stage, which fell foul of complexities of permissions across multiple versions of multiple databases.

6.3.2.4 Address_Create_12092016

Posted April 2016, this SAS script creates an event table for address information from a range of sources. This is no longer needed because of the existence of `data.address_notifications` but it is interesting to see, and it still runs clean out of the box.

6.3.2.5 A_child_parent.sas

Posted in July 2017, this is an adaption of code developed and regularly used by one Ministry's researchers, specifically adapted and shared for another Ministry presumably at their request. It creates a dataset of parent/caregiver relationships to children, drawing on multiple data sources (Census, DIA birth records, MSD) which is clearly an important purpose of general interest. Relationships are difficult to estimate reliably with administrative data and an area of active research for improving the IDI.

This program relies on the existence of global variables like `@IDI_ver` and `@project_dir` and being run within a particular folder system. An analyst can easily change these for their own situation; the script does not point out exactly where and how but the audience of SAS-competent users can be expected to cope easily enough. It assumes that SAS `libnames` have been set up as per the standard Treasury approach, which are set out in (separately distributed) `Std_libs.txt` with code like this:

```
libname dol ODBC dsn=idi_clean_&VERSION.srvprd schema=dol_clean
```

This in turn (unless further adapted) expects `&VERSION` to have been defined, which in the standard Treasury approach is defined in `00_core.sas`, along with a number of other commonly used variables with names like `first_anal_yr`, `last_anal_yr`, `firstage`, `lastage`, `cohort_start` and `start`. These variables reflect a fairly standardised approach for Treasury projects to setting up a cohort for analysis in an evaluative or research project. Running `00_core.sas` in full also would require the analyst to have stored somewhere in their project system other macros defined in SAS program files `CYF_rules_macros.txt`, `BDD_rules_macros.txt`, `Education_rules_macros.txt`, `Correction_rules_macros.txt`, `get_mother.sas`, `get_caregivers.sas` and `Std_macros.txt`. As well as saving those individual files, the user needs to change the paths to them in `00_core.sas` when it calls them via `%include` with an absolute file path.

Once this environment has been set up to mimic the Treasury situation (or various unnecessary parts of `00_core.sas` commented out) and filepaths set appropriately and a SAS `libname` defined for `rel` to hold the results, the code runs cleanly apart from one unimportant frequency table. It produces and saves a range of datasets, of which they key one `child_parent` is about 650MB in size.

This program is a nice example of how having a standard approach to setting up a project can be helpful once obtained. Analysts using the Treasury approach don't need to worry about setting up their environments because they simply call in the same code to do it each time. If they are sharing code with others using the same approach, all is well. Analysts with a slightly different set up have to go to the effort to replicate aspects of the environment that the code was intended for.

The alternatives to this approach are either to:

- Stick to simple, self-sufficient programs. This would effectively limit analysis to the simplest tasks. A collection of modular scripts, with repetitive tasks abstracted into macros and common code stored in one place, is essential for even moderate sized analytical projects.
- Mandate a single uniform standard for `libnames`, key global variables (like which version of the IDI is being used) and some standard macros across the Data Lab community. This doesn't seem to be either practicable or enforceable.
- Wrapping up analytical projects for sharing in fully reproducible containers. This isn't possible with the current technology available in the Data Lab.

6.3.2.6 student performance measure

This file was posted June 2017 by a researcher from outside government and is one of the rare SQL scripts; saved with a `.txt` suffix which gives us more warning about assuming the language a program is written in from its suffix. The aim is to create an individual student performance measure using NCEA data that is as comparable as possible between students - taking into account grades, standards, etc. The code runs cleanly. After tweaking so it put the results into a temporary table (adding `INTO #stud_perf` above the first `FROM` statement) it took about six minutes to execute, generating a table of around 750,000 rows, one student per row, with columns for the various constructed scores.

6.3.2.7 school leavers.sas

Posted in March 2017, this program provides a dataset of school-leavers' start and end dates and qualification codes. It runs fast and cleanly with minimal assumptions (ie that `moe` has been defined by `libname` to connect to the `IDI_Clean.moe_clean` schema, which can be taken for granted for SAS users in the IDI). As a non-specialist in the area it wasn't clear to me what this would be used for; it's possible it was posted to share with someone who had specifically asked for it.

6.3.2.8 MSD benefit (first and second tier) extracts of spells and snapshots for either health condition or mental health condition clients

Posted in May 2017, this code provides datasets on MSD health and disability populations. Four files in total were posted, including the necessary setup files to define `libname` s and make the collection of files self-sufficient. The user just needs to change file paths to point to the user's own folder. Posted with the caveat "the code has not undergone much QA (if any) so please use at own risk".

One of the programs (for mental health spells) runs cleanly until the very last command, which depends on joining to a dataset `home1.general_info` that is a dataset in the researcher's environment. The bulk of the work has been done by this stage and a usable population created with the sought-after data on MSD's mental health flags.

The other program will not run for me as I don't have access to a pre-built dataset `MSD_VOC_DIS_CLIENTS` on the original research project's schema on `IDI_Sandpit`.

This highlights that it is not easy (in fact, not really possible) for one researcher to test whether their code is runnable in other researchers' environments.

6.3.2.9 Māori businesses

Posted in April 2017, this code combines data from the Longitudinal Business Frame with the Annual Enterprise Survey. It is the only code posted in the "Wiki" that relates to the LBD. The program is short, simple and self contained - unlike many of the more complex excerpts posted on the Wiki which have clearly been taken from larger repositories of code with multiple dependencies, this script defines the one `libname` it needs (an ODBC connection to the LBD) and should in principle run end to end. The output is several enterprise-level datasets.

The program runs cleanly, after a part of the code that had been commented out is returned to action (it defines a dataset `fullaeslbf` which is needed later). However, some of the key data is empty; it refer to Annual Enterprise Survey data for year ending March 2015 that has not yet been loaded. Changing a key `WHERE` filter from `201503` to `201403` returns data for tables that otherwise have no rows.

With those modifications, the program runs successfully and retrieves a dataset that could be the beginning of analysis.

6.3.2.10 Virtual Health Information Network catalyst project

There are four files shared in the Data Lab Wiki and on external websites that are products of the Virtual Health Information Network (VHIN) catalyst project and were explicitly designed as exemplar IDI research.

`create VARIANZ population.sas` runs cleanly once the user amends it for their own folder path and name of their schema. This program creates an "estimated resident population" for 31 December 2012, similar to the table `data.est_res_pop` that is now pre-built for all researchers by Stats NZ and gives an indicator of which `snz_uid` individuals are evidenced as part of the resident population on 30 June of each year. It's well structured and clearly commented and does the job of providing working software. It's slow to run; it takes about 30 minutes (would be nice if it warned of this at the top of the program), nearly all of which is in writing a large SAS dataset to the file server (only 2 minutes of CPU time). It could be refactored in SQL, but perhaps 30 minutes is within acceptable bounds for a once-off data preparation step.

`create VARIANZ dataset.sas` adds a range of demographic, geographic and health variables to the population. It includes the useful `overseasdays()` macro to create a table with days spent overseas for each person. It takes ethnicity from census if available, otherwise from health, otherwsie from personal detail table - either this program precedes the ranked source ethnicity table or the researchers made a deliberate decision not to use it.

To run this program, the analyst has to first create an additional table called `address_subset` in their `IDI_Sandpit`. Comments in the program state that

> *"Using SQL mgmt server (SAS is too slow) I have already created a subset of address notification table with 'MSDP' type addresses removed, and updates after 31 Dec 2012 removed".*

The SQL that does this is not supplied as far as I can tell; obviously it is easy enough to re-develop but in the interests of reproducibility it would be better if it came with the SAS program. So, I think what was needed is this:

```
SELECT *
    INTO IDI_Sandpit.[DL-IMR2017-05].address_subset
    FROM IDI_Clean.data.address_notification
    WHERE ant_address_source_code <> 'MSDP'
    AND ant_notification_date <= '2012-12-31';

ALTER TABLE IDI_Sandpit.[DL-IMR2017-05].address_subset
    ADD PRIMARY KEY (snz_uid, ant_notification_date);
```

… but I can't be sure I've done it exactly the same as the original author. This lets the program run past the point it previously failed, then it fails when it can't find a dataset `areas_file_2013` in a folder where it was meant to have been placed previously. It might be that there is another program somewhere that creates the `areas_file_2013` file, but it's not clear where I should look to get it.

It is not meant as a criticism but an observation that the the culture of "out of the box" reproducible research hasn't infused the IDI community. The practical difficulty in sharing a collection of scripts is surely one of the main impediments in this area. The Sharepoint discussion board is a clumsy way of sharing computer programs and this becomes particularly obvious when there are several files or a whole folder system to share (as will increasingly be the case for anything of even modest complexity).

The minor issues above are now being addressed, but with the permission of the researcher responsible I am leaving the section above as-is, as it's a good example of the challenges with the current code sharing environment.

6.3.2.11 Code files for Stats NZ Otago youth NEET pilot partnership project

Posted in July 2017, this project shared eleven beautifully organised and commented SQL files, in numbered order. The elements that need to be changed (ie location of a schema the researcher can write to in `IDI_Sandpit`) are clearly marked and the code runs smoothly, reproducing the analysis end to end (other than random rounding for confidentialisation and output release).

6.3.3 Social Investment Agency

Caveat for this section: I was Lead / General Manager Evidence and Insights at the Social Investment Unit / Agency while some of the material mentioned below was developed or published.

The Social Investment Agency publishes code on GitHub. This has the great advantages of:

- being visible to non-IDI users
- allowing more complex repositories of code, including sub-folder systems, to be shared
- making versions and releases trackable
- presenting in a platform designed for sharing code (so with syntax highlighting, line numbers, etc).

The disadvantages are three:

- researchers wishing to use it need to follow more steps than for code shared in the Data Lab Discussion Board (ie they need to download a zip file and ask Stats NZ to drop it into their research folder);
- the code is not presented together with the larger body of code on the Discussion Board, hence reducing findability for researchers
- SIA has an increased maintenance burden from keeping their development and production versions of code in the Data Lab in sync with the external (and never run directly) version on GitHub.

In addition to one end-to-end analytical project (on social housing (https://github.com/nz-social-investment-agency/social_housing)), it has provided two significant pieces of functionality to build persistent data assets in `IDI_Sandpit`:

- the Social Investment Analytical Layer or SIAL (https://github.com/nz-social-investment-agency/social_investment_analytical_layer), which creates views and tables of events in individuals' lives following standard naming and structuring conventions, and joined to average cost information which lets analysts easily derive approximate costs incurred by individuals
- the Mental Health and Addictions definition (https://github.com/nz-social-investment-agency/mha_data_definition) which builds persistent tables for all individuals in the IDI spine indicating their interaction with mental health services

The third main contribution for re-use is the Data Foundation (https://github.com/nz-social-investment-agency/social_investment_data_foundation), a portable project to build a dataset suitable for analysing the life path of individuals at regular intervals for a certain period before and a certain period after a particular event (eg an intervention in their lives by the government). Unlike the SIAL, this project does not aim to build a common *dataset* that is identical for everyone who runs it, but provides parameterisable *functionality*.

The SIAL is inherently something that only needs to be created once pre refresh of the IDI, and now it has been incorporated by Stats NZ into the IDI refresh process and included in the new `RnD` database. Unfortunately due to a miscommunication between the two agencies, in doing so the tables and views have been given date-stamped names which is poor practice and makes it very difficult to maintain code over time. For example, the values of `snz_uid` in the date stamped version of the SIAL will soon not concord with the `snz_uid in IDI_Clean`, so code that works correctly today linking the two will (silently) give nonsensical results after the next IDI refresh. A better approach, pragmatically working with the current environment, would have been to include the SIAL (and also all the content of `IDI_Metadata`, which has the same issues) in `IDI_Clean` itself so it is refreshed and archived in sync with the main database.

The Data Foundation is closer in spirit to previous IDI code sharing in that it provides functionality for an analyst to adapt rather than a persistent widely-used data asset (even though the end product is such an asset, it is specific to the analytical question at hand). An analyst wishing to use it must first have built their own version of the SIAL (the Stats NZ version won't work because of the changed names), and then modifies a copy of the main parameters of the control script to reflect the particular population they are interested in. It draws on a library of SAS macros that are intended to be stable, used for multiple projects, and not touched except for maintenance by the SIA.

The SIA's published code and documentation is to a high standard and they are unique (as far as I can tell) for IDI researchers in having an organisational coding style guide that covers SAS, SQL and R code, rather than relying on individuals to set their own standards. Documentation is explicitly aimed at external users at a lower level of knowledge than the developers (as it should be), and the code is designed to be as portable as possible. Unfortunately, this still isn't very portable by standards expected outside the Data Lab.

The SIAL code runs smoothly. The data foundation is newer and more problematic, but also now runs after fixes made by SIA during the writing of this report. Both require on-going maintenance from SIA and are vulnerable to any instability in the IDI refresh process - for example, changes in the `IDI_Metadata` database which are sometimes undocumented, changes in column types due to the manual nature of the refrsh process, etc.

The SIA's approach is very ambitious in aiming to provide functionality that does more complex things under the hood than are visible to the intended users. This is an innovation for the IDI but is standard practice in the outside analytical world. Unfortunately, several obstacles stand in the way of success in this regard:

- As an overall statistical computing environment, SAS is not friendly to extension. Unlike Stata and R which are designed for user-contributed enhancements in the form of libraries or packages, SAS does not have a built-in system for managing new packages of functionality, their helpfiles, and the dependencies between them. Providing source code of macros, asking users to make manual adjustments to file paths, and to carefully manage the locations of macros in their own system is a poor cousin of typing `install.packages("sial")` which is what an R or Stata user is used to. The inability to build on the helpfile system is a particular disadvantage.
- The database itself is unstable; rather than having a persistent design with new data inserted into it, it is rebuilt from scratch each quarter and small changes, sometimes undocumented, creep in, which lead to revisions being needed regularly to existing software.
- While SIA has made big efforts for their software to be as "plug and play" as possible, it still represents an increase in scale from other code being shared (for example, by requiring whole folder structures be copied, not just individual scripts). While this is the norm in analytical projects in the outside world (where a large project might be re-created locally with a simple call to `git clone`) it is unfamiliar and daunting to many IDI researchers. When things go wrong it is harder to troubleshoot such a system than a simple script.
- The problems above are greatly exacerbated by the lack of version control in the Data Lab, as is the development of SIA's programs in the first place. Serious software development cannot take place without source version control, and the SIAL and Data Foundation are beyond the limits for what is safe in terms of complexity.

Around 10 research projects have imported the SIAL source code and I found evidence some of them are using it. Other researchers interviewed reported looking with interest at the SIAL and using some of the code defining the views, rather than changing their workflow to have queries pointing at the SIAL itself. I did not find evidence of any researchers using the "Data Foundation" functionality yet.

6.3.4 "DL Common Files" or "Data Lab Resources"

Predating the Wiki, a directory on the file server `wprdfs08` named `DL Common Files` has some common resources including

- two Excel workbooks `excel_grr.xls` and `random_round_rr3.xlsm` with random rounding macros for use in confidentialising output
- SAS macros for rounding for confidentialisation purposes:
- four for graduated random rounding (`GRR.sas` , `GRR_for_LBD.sas` , `GRR10.sas` , `GRR1000.sas`)
- one for base three random rounding (`RR3.sas`)
- one for "unweighted counts action 1b" which means unweighted counts of 1 to 5 are rounded to 5, 0 stays at 0, and all other counts are rounded to the nearest 5
- a SAS macro `pconfid.sas` for checking confidentiality and producing variable totals for a value variable
- an R function rrn for random rounding for an user-provided base. Unfortunately this is in a `.txt` file and the first two lines of (non-R) have not been commented out, so it cannot be called by `source()` without editing.
- a SQL program `Pent_RF.sql` , described as a competition project for NZAE conference 2012 "to extract Pent (and Ent) from RF" for the LBD which [does not work] / [no longer works] because it looks for a column `leed_employee_count_nbr` in the `dbo.load_lbf_fact_pbn_employee_count` table that does not exist.
- `Programs S Dixon.zip1
- a library of Stata ADO files

These rounding SAS macros are some of the most copied bits of code as around half the projects have made their own copy. Obviously this means centralised updates or maintenance is difficult. The `REAMDE.txt` file instructs users to make their own copy, so researchers are following instructions.

All these macros are very widely copied by research projects. From interviews, the most widespread tool for random rounding to confidentialise output is the Excel macro. Some researchers find this approach transparent, and easier than a coding solution. There would be some efficiency gains from using the SAS or R macros for random rounding (and some projects make the most of these) but these are not particularly decisive, given most output needs to be placed in Excel for output-checking purposes anyway.

There are also Stata files for random rounding but they circulate by hand rather than stored in this or another central location.

In browsing through researchers' folders, I found five different random rounding R functions, including three variants of one by myself (two variants by me, one by another researcher) and two by the author of the one in the `DL Common Files` folder.

6.3.5 The LBD "codebook"

Commonly used code from the early years of analysis of the LBD has been collated into a "codebook" which is passed hand to hand by researchers in the know.

7 Opportunities

My fifth research question was "what opportunities exist to make life easier for researchers, and to capitalise on the gains they've made so far?" This report is not in a position to put forward concrete recommendations. However, I did identify a number of opportunities that could be considered by the relevant people. These observations can be considered along other priorities raised through the rest of the IDI 2 project (eg under the "Access Pathways"" workstream) and the existing product backlog. These opportunities fall into four main categories:

- things Stats NZ could do, fairly quickly in principle if resources permitted, that would improve the analytical environment, and in most cases could be referred to the "Access Pathways" workstream of IDI 2 or dealt with under business as usual
- more substantive issues for Stats NZ to think about during the "Fundamental Redesign" workstream of IDI 2
- changes to practices that researchers might want to consider
- issues for me to consider in my options paper on "layers" for the IDI

7.1 Things for Stats NZ to consider that would improve the analytical environment

These suggestions are ranked in order of importance as I see them.

7.1.1 Version control software

No analytical environment without version control software can be considered to meet professional standards. The lack of a tool like Git or SubVersion puts sharp constraints on the complexity and scale that could be achieved with the IDI and LBD. Addressing this should be very high on the priority list for Stats NZ.

In addition to the primary purpose of helping projects manage their own repositories of code, version control software would be part of the solution for better sharing of code. The platform under Stats NZ's consideration, GitLab, facilitates repositories to be cloned and forked, pull requests, and other functionality to support collaborative development. Care should be taken that its implementation in the Data Lab makes use of these features. Researchers need to be able to clone eachothers' work, and refer to repositories of code directly, within the Data Lab (when code has been marked for sharing, of course). This is simply a logical extension of the current potential via the Discussion Board of the "Wiki", but in a tool designed for code sharing in a way that Sharepoint is not.

The rollout of GitLab will need to be accompanied by training, or at least guidelines on its use. Those guidelines should include (amongst other things):

- researchers should not commit created data objects (eg sas7bdat, dta, rda or csv files) to the repositories - for both security and size constraints (guideline should show how to use a `.gitignore` file to facilitate this)
- researchers should not commit ouput objects (eg csv, Excel, images, HTML, etc) - for the same reasons
- repositories that are "public" to other data users must particularly not include any data, for confidentiality reasons
- suggestions on how to use multiple repositories, to use repositories for team workstreams not individuals, but for individuals to have the ability to work on their own clones of a team repository.

7.1.2 Database tuning

The databases holding the IDI and the LBD have been neither designed nor tuned for performance in extracting data. Many of the tables in the IDI in particular do not have primary keys, and columns that are commonly used as foreign keys in joins nearly all lack indexes. The LBD has better indexing but still opportunities as seen in Annex D. The design is non-trivial to change, but there are almost certainly low hanging fruit on tuning and indexing that would deliver big improvements in speed.

Resources and time permitting, the Stats NZ Digital Business Services team, which includes the database administrators (DBAs) for the servers in the Data Lab, have indicated they could work with Integrated Data to tune the database to make queries much faster. It would be a fairly straightforward to identify a range of 50 - 200 representative queries and tune the existing databases via indexing and other tools they have in their toolkit.

7.1.3 More (and better integrated) "metadata"

Much of the data commonly called "metadata" consists of lookup tables linking codes to items such as the text of survey questions, full names of pharmaceuticals and ethnicity names. In warehousing terms, this is just part of the data that should routinely be included in the same layer as part of dimension tables.

There are issues with the current way this is stored in the IDI that lead to problems when classifications change, but these are issues to consider in the redesign. In the meantime, Stats NZ could add significant value by addressing gaps in the metadata in the databases (ie as tables that can be joined with SQL, not just document collateral). A key example in this space is the LBD survey questionnaires, which should exist as tables linking both question and answer codes to their full text; this would not be a particularly large job (although it is non-trivial precisely because the questionairres are not currently in easy machine-readable shape) and it is a one-off investment that would make the data much more usable.

7.1.4 Re-orient Stats NZ documentation around SQL rather than SAS, R and Stata

Researchers will use the combination of languages they are most comfortable with for their own work, and these choices should continue to be up to them and their teams (obviously, there are big gains for teams to agree an approach for which languages to use for which purposes). However, when it comes to training, sharing code, and in particular to "productionising" any standard approaches, more emphasis should be given to SQL, the lingua franca for relational databases.

There is a common misperception that the toolkit for the IDI is a choice of statistical packages. Hence we are seen to have a large SAS community and a smaller R and Stata communities. In fact, the choice of statistical package to use is very secondary. The important thing is that researchers need to start their analysis with SQL, and their choice is whether to do this in pure SQL or in SQL in combination with SAS. The big bulk of work in the Data Lab is data preparation; only rarely does the choice of econometric or statistical model and its implementation in SAS, Stata or R dominate workloads.

The misperception is subtle. The Stats NZ website states accurately:

> *"If you intend to use the IDI or LBD, we strongly recommend having intermediate SQL coding skills. Most researchers code in SAS or SQL, while others prefer coding in R or Stata." (Stats NZ website (http://www.stats.govt.nz/tools_and_services/microdata-access/data-lab.aspx))*

… but the emphasis could be changed. For many reasons, the workflow of researchers should be reconceptualised as three steps characterised by three different tools:

- SQL to get the data together in the database and extract it into a statistical tool
- One of SAS, Stata or R to perform statistical analysis (this step can be avoided altogether if one is just doing frequency counts and cross tabs, although this is rare)
- Excel to prepare the data for output checking (this last step can be wrapped into the analysis step by those who write SAS and R programs to do the confidentialisation, but these people are in a minority)

This is how LBD researchers behave, but not those in the IDI. The use of SAS to wrap `PROC SQL` around SQL obscures the importance of SQL as people then talk of SAS programs, when they are in fact a combination of SAS and SQL (often including more SQL than SAS). Much of the code that has been shared in SAS could relatively easily be made much more multi-lingual by extracting the SQL components, and turning other data tasks done with a `DATA` step (and hence conducted on the SAS server) into SQL. The advantages would be both efficiency and cross-language portability. The Stats NZ Otago Youth NEET project provides an example in this space.

SQL is *the* language for relational databases; its mastery is highly desirable if people are to understand the data. It should be given priority in publicity, sharing of code, and training material. This won't happen overnight, but we should move in this direction. Subtle signals could be given, including re-writing the excerpt on the Stats NZ website as:

> *"If you intend to use the IDI or LBD, we strongly recommend having intermediate SQL coding skills. After data is merged, filtered, sometimes aggregated, and extracted with SQL, most researchers then complete their analysis in one of SAS, R or Stata, and it is strongly recommended you have skills in one of those three languages as well as SQL. Some SAS users use only minimal SQL, but this is not an option for R or Stata users." (proposed re-wording for website)*

Having some SQL skill and using SQL more is helpful to researchers themselves (because their code will often run faster, and because it forces them to work with the data structures as they are actually set out in the database) and helpful to other users (because it doesn't slow the system down so much).

It is possible that with more efficient queries written in SQL rather than SAS and executed on the database server, there will be less need for researchers to save intermediate data objects because they know they can re-create them easily from code. Version control would help here too (as they will be assured they can recover exactly the same version of code).

7.1.5 A 'researcher's rough guide to the IDI', buddy system and training course

Several researchers lamented the lack of a researcher-oriented manual or guide to the IDI comparable to Fabling and Sanderson's *Rough Guide to the LBD*; or the lack of an introductory training course (preferences vary).

The need for a researcher-oriented guide is widely acknowledged including by Stats NZ and is being considered as part of IDI 2. Some of the material in my report could be repurposed for such a guide, as could the eight existing introductions that have been collated by Stats NZ, and their own first draft of a guide.

Given the diversity of IDI researchers and potential researchers, a course could be difficult to pitch. There are at least two different types of researchers needing support:

- those who are already skilled in SQL and one of SAS, Stata or R, and just need introduction to the data and its metadata
- those who need to build skills in the toolkit along with understanding of the data

Obviously, there is a continuum along these dimensions. The "six months to learn to use the IDI" figure I have heard would only apply to users in the second category, who naturally have a longer learning curve to negotiate. New researchers with the toolkit under the belt at the starting point have successfully turned around reasonably complex projects within two to three months.

It would probably be worthwhile to develop courses oriented to both target audiences, but arguably only the first type of audience is a role for Stats NZ - depending on the interpretation of its Data Leadership role. For the second audience, there are already training courses available off-the-shelf on SAS, R and SQL although none of them (to my knowledge) exactly fit the niche. Stats NZ or some other group could consider adapting such courses to particular analytical challenges of the IDI, if it was thought this was an appropriate role to provide such training.

Training on the IDI and LBD databases themselves should focus on choosing, joining and filtering data and metadata in SQL Server Management Studio where the users get the most direct experience and understanding of the databases. It should be agnostic to choice of statistical tool between SAS, R and Stata, and in fact probably contain very little use of any of those tools other than the basics of how to connect them to the database and use data that has been put together with SQL.

Researchers reported that in the early days of the LBD a semi-formal buddy system was fairly successful in introducing new users to the complexity of the data and what can be done with it. A buddy system can scale up in ways that centralised training can't. Stats NZ could consider setting up systems to facilitate this (or of course, it could evolve spontaneously, as in fact is the case in larger organisations and universities).

7.1.6 Publish style guides

It is not up to Stats NZ to dictate researchers' coding style, but there is a valid role in setting forward a good practice standard. Many researchers in a diverse range of projects are writing high quality code, and some others are willing to learn but have less experience of the discipline necessitated by analytical development taking place in teams. Promulgating good practice style guides would be low cost and help raise standards of coding and hence efficient and effective data analysis.

Internally, Stats NZ has a good quality SAS style guide, although perhaps a little dated (it is silent on the best use of Enterprise Guide projects, for example). This guide or its updated version should be published as an example that externals could choose to follow or adapt, and similar documents developed and published for SQL and R. As there are plenty of public examples in these languages, the original work required should be minimal. Stats NZ is aware of this need, and meeting it is a matter of timing and resources. There is interest in this issue beyond just the IDI and LBD and a range of existing work to tap into; for example the Social Investment Agency has a style guide for SAS, R and SQL in the IDI that could be adapted or adopted.

As far as I am aware Stats NZ has few Stata users compared to its users of SQL, SAS and R so it may not be in the best position to develop a Stata style guide. This might become a community project for researchers. The existence and publicity of style guides or a combined style guide for SAS, SQL and R (which would have many features in common) would doubtless help anyway.

7.1.7 Review the Sandpit and the LBD Researcher databases policy

The `IDI_Sandpit` is not operating the way the policy describes. I believe there would be gains in encouraging researchers to use it in the way the policy describes but permissions do not allow sharing data between projects with access to the appropriate schemas. So at a minimum, permissions should be changed to make that possible as per the policy.

But I think there are even bigger gains from encouraging researchers to save their intermediate data objects in the Sandpit rather than on the SAS server. It is much more efficient to copy data from `IDI_Clean` to `IDI_Sandpit` than to the `wprdfs08` file server. This is partly because it is the same machine, and partly because SQL Server is designed to handle such large amounts of data whereas this is not the natural function of the file server. This change would have implications for Stats NZ's infrastructure.

A housekeeping issue related to this is the desirability for the separate uses of the Sandpit to be more separate. For example, tidy datasets for sharing with multiple research projects could be kept separate fromgenuine sandpit working areas. Several researchers described the Sandpit as a "mess" and difficult to work with.

The review of the Sandpit and the LBD Researcher databases might make most sense as part of the "fundamental redesign" workstream of IDI 2.

7.1.8 Other documentation

Researchers report being overwhelmed by the information and the main media in which it is provided, yet it is clear that more is still needed. The trick is to do this in a way that will be be more consolidated, searchable and in a small number of spots, as close to the data itself.

Opportunities exist for better documentation for some of the excellent developments in derived tables and metadata in recent years. For example, as one researcher writes:

> *It would really help users if there was more information about the derived tables (source_ranked_ethnicity, address_notification, person_overseas_spell, etc). There is a small amount on the website, but it's separate from the other metadata and it doesn't contain enough detail to understand how they were created. Same comment for the tables in the metadata database- it's really a guessing game as to what some of them contain. In some cases, I think people are importing files and storing them in their file folders when they don't need to (I know I have done this before). A good example is meshblock concordances- there's one in metadata, but a lot of people don't know it is there, so they import their own concordances.*

7.1.9 Whitelist H2O and install it on `rstudio03`

H2O (https://www.h2o.ai) is the world's leading open source deep learning platform and used by over 100,000 data scientists and 10,000 organisations around the world. It is designed to make the most of a cluster of servers which would entail significant architectural redesign in the Data Lab; but it can also operate on a single machine.

Installation on the single `rstudio03` server would give access to its state of the art powerful, optimally written Java programs that successfully fit a wide range of models to very large data, including generalized linear models with elastic net regularisation, random forests, and deep learning neural networks. The software is open source (ie free) and basically just needs Stats NZ whitelisting for installation. It can be installed and accessed via an R package, and from the researcher's perspective would be just another piece of R functionality. It is currently blocked by security permissions because installation requires downloading some `.jar` files from Amazon. The security risk could be assessed by Stats NZ and, assuming it passes, H2O installed in the central R package library for all users. H2O is available for at least one other government department (MBIE).

This would not make performance problems related to computationally intensive statistics vanish, but it would be a powerful additional tool for researchers facing challenges of fitting statistical and machine models to large sparse data such as sometimes occurs in the IDI.

7.2 Things to think about for the "Fundamental Redesign"

7.2.1 Building on a persistent version rather than building from scratch

Unlike the LBD, the IDI is built from scratch each refresh process. If it could be redesigned so the new data added to the existing version rather than creating from scratch, there would be a number of advantages. From a perspective of how the presentation layer is seen by researchers, chief of these advantages is that it would be possible to permanently attach a random seed to each individual in the database. This is needed for any of the variants of automated confidentialisation that Stats NZ is considering. However, moving to a persistent `snz_uid` would be a non-trivial change and considerable difficulties would need to be worked through in the design process, such as management of "individuals" that were found to be wrongly linked.

7.2.2 How much cleaning to do?

Significant complications for researchers' work involve dealing with contradictions within the data such as:

- duplicate entries that should be impossible (eg the same `snz_uid` featuring twice in the `dia_clean.births` table)
- overlapping spells that should be impossible (eg when the administrative process only allows people to be in one school/prison/beneficiary category etc at once, but the data has then in two).

These problems come either from the Stats NZ matching process or, more commonly, from the original data from contributing organisations. In the redesign, consideration should be given as to how much Stats NZ should address this during the loading process, whether by cleaning it up in the process or rejecting and requiring the source agency to clean it up. In the meantime, this sort of challenge forms a likely candidate for some kind of centralised functionality (eg a centrally agreed way of addressing overlapping spells, building on one of the methods circulating in the form of SAS macros).

Related to this is the possible need for more standardisation in column and table names.

7.2.3 More normalisation and possibly also more dimensional modelling

The wide tables used for storing survey data are difficult to work with and are probably inefficient to query on the database (although with modern developments in databases this may be less of an issue in the future). Deciding on how to present this data to researchers will be an important question to consider in the redesign.

The extensive use of TRUE/FALSE or 1/0 in dimension tables is unhelpful for efficient analysis and I think could be improved by using meaningful text that would engage meaningfully with drop down filter tools or grouping levels out of SQL. Annex C works through a situation where a bit more normalisation of an IDI table in combination with meaningful names for classifications currently referred to by numbers makes analysis easier and possibly quicker for system performance. But even beginning to think about this opens cans of worms…

I'm not sure if it is important to create a fully normalised version of the data at some point, and on whether a formal dimensional model would be helpful for presentation to researchers, but I am confident that *some* move in one or both of those directions would be helpful, not just for the existing type of researchers but for others who could benefit from a Business Intelligence tool that would be more effective on top of a more formally modelled data structure..

Some of the complexity of the IDI data model comes from the need to grant researchers permissions only to part of the overall database at once. If the redesign includes a move to a platform such as SQL Server 2016 (in contrast to the 2012 version currently used), the permissions model could be moved to a row-wise (https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/granting-row-level-permissions-in-sql-server) basis. This would allow a data model to be developed based on researcher needs and free up a number of current constraints.

7.2.4 Treat the "metadata" as part of the data

The history of what researchers and Stats NZ call "metadata" is one of slow improvement, but the current state reflects the history. The `IDI_Metadata` database is clearly seen as a repository of information that is somehow external to the data itself - reflecting a history when it originally existed in intractable documents, then hard coded as `FORMATS` in SAS programs. A data warehousing approach would see this "metadata" simple as dimensions of the main data, and the warehousing job is not complete until it is formally linked together.

The extreme of this situation comes with the LBD, where there is no machine-readable version of the questions of the surveys and Inland Revenue forms that make up much of the data; but there are still issues related to this in the IDI too. The redesign should carefully think this through, and the dimensions currently held in the `IDI_Metadata` database need to be better joined to `IDI_Clean`, if only to avoid problems such as I found in my mini-case studies of analysis that no longer works due to the two databases getting out of sync.

7.2.5 Better treatment of slowly changing dimensions

Related to the above problem, a solution is needed to handle slowly changing dimensions. In addition to the example mentioned above with Justice reoffending, several researchers raised challenges with meshblocks. There are standard data warehousing solutions to such problems, which should be considered during the redesign.

7.3 Things researchers could consider

Researchers are achieving great things with the IDI and the LBD, and I am acutely aware they have (mostly) not asked me for my suggestions. So the following are suggested tentatively for their consideration.

7.3.1 Use more SQL

Researchers will always have their own preferred tool, but from examining code and speaking to people I think many could benefit from spending more time in SQL Server Management Studio developing their code. The selling point is efficiency of doing as much as possible on the database server, and the specialist environment Management Studio provides that is dedicated to precisely this task of exploring and understanding databases and joining, filtering and aggregating data in them.

7.3.2 Share more code, and interact more on theDiscussion Board

Many researchers were sceptical of the benefits of sharing code on the Discussion Board on the Wiki, but I was pleasantly surprised by the quantity and quality there. It would form an excellent introduction for new users to work through examples of that code. Many research projects have other good code that could be shared and they should do so. Although the Discussion Board is not the best tool for the job, it does work when it comes down to it.

7.3.3 Use subfolders more strategically

Better use could be made of subfolders or at least naming conventions to differentiate between input data, intermediate data, output, and analytical code. This then facilitates clearer thinking and easier communication about dependencies within the analytical program. The Microsoft Team Data Science Process, already mentioned, has some good principles on this.

7.3.4 Coding style

Some researchers are writing very high quality code with clear documentation, good flows, and even refactoring for efficiency in some cases. Senior researchers and managers should send signals to their teams, and particularly to new researchers, about the importance for good coding style for working in teams. All projects have some kind of peer review; coding transparency and style should be added to the lists of things reviewers examine (where this isn't already the case).

The most common improvement I thought could be implemented in terms of coding style related to documentation and commenting. Documentation could be improved by stating more of the *why* things are being done rather than the *what*. This applies both at a strategic level setting out the overall approach (either in README.txt files or in a description at the top of the first script to be run) and in in-line comments.

7.3.5 Do more confidentialisation in R and SAS

Both SAS and R have macros and functions that do random rounding (base n or graduated random rounding). R also has the powerful `sdcTable` and `easySdcTable` packages for a broader range of statistical disclosure control including p value suppression and secondary suppression as required by Stats NZ. Researchers could consider making more of this functionality.

7.4 Issues to in considering options for "Layers"

The IDI and LBD are presentation layer datamarts lacking a foundation, rather than a foundation lacking a layer. But this is something that can be improved on in the future. These are things for me to consider while thinking about options and high level design for "layers" in the IDI:

7.4.1 Persistent data objects to help researchers

I found few instances of data that is created by researchers themselve that are likely to be of broad use across other projects. Nearly all projects created intermediate data objects for their own purposes but these are mostly of interest only to themselves.

7.4.1.1 New centralised tables or views?

There are a number of tables that would benefit from being created centrally and used by all researchers. "Spells as a NEET" is one example, as might be a more consolidated "Spells as a beneficiary" (possibly already fixed in the SIAL).

The most likely candidate for central creation and control was MBIE's "spells" dataset, which researchers into migration issues create from the IDI each refresh process and is used as a starting point for many ad hoc queries and standard reports in MBIE. While most of the researchers who would be interested in this already have access at MBIE, there would be some additional interest, and it might be safer and simpler for Stats NZ to take over the creation of this dataset each refresh.

There are also some minor enhancements to existing tables possible. For example, there is a "tax paid" column in a dataset available in `IDI_Sandpit` that is not copied over to the `IDI_Clean.data.income_tax_yr_summary` .

7.4.1.2 The SIAL

The SIAL is the other obvious candidate, and steps have already been taken to centralise it's building on a quarterly basis. The thing sthat make the SIAL useful are:

- standardised data structure and naming conventions
- standardised classification and roll-up of some variables eg benefits.
- combination of the events data in the original IDI with supplementary estimated costing information

The first of these challenges should not be up to something like the SIAL to fix, but should be solved by a consistent approach to data modelling in the original database design. After the "Fundamental Redesign", that aspect should be redundant.

The third of the benefits listed above poses an interesting question. From a warehousing perspective, the costs of various "events" (eg being on a benefit, being in prison, being at school) is slowly changing dimension information. It should be in the main database - as should similar dimension data currently in `IDI_Metadata` - not separated out in the `RnD` database with the SIAL. Another thing to consider in the redesign.

As mentioned in the main body of this report, due to a miscommunication there are problems with the actual centralised implementation of the SIAL. Tables' and views' names should not include date stamps in their titles, and this habit (seen with the SIAL and in some tables in `IDI_Metadata` as a way of handling slowly changing dimensions) should be nipped in the bud.

7.4.1.3 Census Transformation by-products

The most promising area in this space is to build on the very popular (with researchers) success of summary tables in the `IDI_Clean.data` schema such as

- estimated resident population
- summary tax information
- ethnicity ranked by source
- address notifications combined by source

Stats NZ's Census Transformation project is the most likely source of such additional tables. They are particularly useful when they can provide a single solution for a common problem eg "of all the conflicting sources of information on ethnicity, which should I use?". Ultimately, delivering such tables for all ambiguous areas would leave analysis of the IDI much simpler than it is. Thought should be given to how to make the most of this.

The aims of users interested in estimating population statistics from the IDI basically coincide with the Census Transformation project, although they are coming at it from opposite ends.

A set of weights, which would on average by slightly less than 1, for individuals estimated to be residents of New Zealand, would be a good additional variable for the key dimension table `data.personal_detail` . A definitive random seed, to be used in automated confidentialisation when methods for this are determined, would be another obvious addition. This seed would need to be the same for individuals across multiple refreshes, which poses problems given all people get a new value of `snz_uid` each refresh; this might be an issue to kick to touch for the Fundamental Redesign.

Many researchers are keenly interested in a standard way to analyse households and other relationships in the IDI.

None of these issues are simple to solve; all need serious consideration and methodological analysis before a solution is found, if it is possible at all. These issues are on the Census Transformation work program and as solutions emerge should be integrated into the IDI proper in the same way previous developments have been.

7.4.1.4 And in the LBD?

The LBD is different. Significant additional work is needed to make the LBD analysis-ready. While it is annoying that (for example) some columns in the IDI are DATE and some are DATETIME, the challenges with the LBD are of another order and relate to important parts of it such as the Business Operations Survey simply not having been transformed ready for analysis. However, this isn't a question of another layer being needed, so much as designing a good data model and finishing the development of its extract-Ttransform-load systems. I wouldn't recommend creating a layer as a hack to paper over problems in the underlying data model. So work should continue (as it is) on improving aspects of the LBD which are not analysis-friendly; and even if the LBD is not included in the scope of the "Fundamental Redesign" of the IDI, its issues (including how much further we should go towards a formal dimensional model for integrated data - the LBD goes further than the IDI on this in some ways, less in ohters) should be considered in that redesign.

7.4.2 Shared functionality

While there are no obvious game-changers in terms of creating new views or other datasets within the IDI, there are some probable wins in terms of creating shared functionality. The prevalence of numbers of pieces of shared code and numerous shared problems suggests that some common problems should be developed and maintained centrally - whether by Stats NZ, or by the researcher community open-source style, facilitated by Stats NZ.

Opportunities include functionality such as:

- determining meshblock most resided in by calendar year
- dealing with overlapping "spells" in a consistent manner (for example, students enrolled in multiple schools at once, or prisoners in multiple prisons)
- reoffending
- spells on benefits, rolled up in various ways (a decision would be needed on how this compared to the original data, and the version in the SIAL, both of which provide a version of this)
- amount of time spent overseas between any two dates
- return estimated resident population on any arbitrary point of time, possibly with weights that add up to the official statistic for that point of time.

and there are doubtless more. Some of these are simple operations, some are more complex. Some might be better created as static tables rather than functionality; principles and criteria for deciding this would need to be determined. The key issue would be to agree with relevant data providers the definitive *standard* way (which could be over-ridden if analysts need to) to interpret and roll up their data, and implement that method once.

The best mode for delivery for such functionality would be a centralised library of SQL, perhaps as stored procedures or possibly as some other technology, which could run on the database and would have light front ends in SAS, R and Stata. The SQL would only have to be maintained in one spot. The SAS functions would be stored centrally where everyone can access them without copying code; the R and Stata functions would be R packages and Stata libraries also available centrally. Those libraries would also include definitive versions of the various random rounding and other macros and functions, which are currently not maintainable due to the distribution model.

Considering the way forward on this possibility is definitely a job to be done in thinking of options for the "layers" project.

7.4.3 Other things to think about…

Some miscellaneous points that don't really fit anywhere else. There's no rush to consider these but they are worth mentioning:

- As a complement to the existing languages of SQL, SAS, R and Stata, it might be useful to get a more downstream modern business intelligence tool like Power BI, Tableau or SAS Visual Analytics into the Data Lab. Several researchers have mentioned this. It is something that might be useful directly if there were a more formal data model in place eg a dimensional model star schema, which plays nicely with modern BI tools. Alternatively, it would just be another tool for researchers during their exploration and dissemination phases
- The Data Lab is missing a general text editor like Notepad++ or Atom. Notepad++ is available in the standard Stats NZ environment so it would presumably be straightforward to make it available in the Data Lab too. At the moment, editing of text files has to be done in RStudio, SAS EG, or Stata (or windows Notepad…), which is a productivity hit for researchers used to more powerful tools.
- It would be nice to have a powerful Q&A tool (Stack Exchange style) available in the Data Lab; one that allows voting for questions and answers, easier formatting, easy to insert snippets of code. Such tools are available. This would be much better than the existing Discussion Board and it would be great for it to be considered as part of a strategy to create a real analytical community. But it is down the priority list, particularly compared to issues such as version control and database performance. Implementation of version control could be a game-changer in itself for creation of community and might mean less need for such a Q&A platform.

8 Annex A - Method

The research behind this report involved:

- defining the good practice framework set out in section 2.3, based on an informal survey of the literature and my own experience
- nine interviews with eleven researchers, many of whom also generously provided follow-up information via email and answered my numerous annoying questions
- observation and analysis of the databases themselves. This was used for everything from determining the number of tables and sizes of the databases, to observing the queries that took the longest time to run. Most of this analysis was done directly via standard reports that ship with SQL Server Management Studio; some more customised things were written directly in SQL
- the filenames, full directory paths and file sizes in four different locations for researchers' project folders were piped from various calls to `dir` into eight text files and analysed with R
- code that was published on the "Wiki" Discussion Board was adapted and re-run
- researchers code was examined (with their permission - see Annex E). An overview skimming review was made of many researchers' projects, then 11 were examined in more detail and four case studies produced in Annex E of this report. Researchers' folders were also invaluable in following up in a qualitative way questions about issues such as folder structures, how they handle version control, repeats of files, and how code shared on the Wiki is re-used. Researchers' requests to have their code *not* looked at were strictly observed.

Parts of this report are based on analysis of metadata in researchers' folders. This is the researchers' work saved to a network drive; usually `wprdfs08` (in the MAA folder system for external researchers or IMR folder system for Stats NZ researchers) but for some users on `wprdsas10` (Treasury) or `artsas01` (MoJ, MSD, MVCOT). See the diagram at the beginning of the report to understand how these research folders relate to the user's virtual PC, the SAS and RStudio servers, and the actual database servers.

The information available is the names of files and folders, their suffixes, and sizes on disk. To avoid any sensitivities, research project names have been removed.

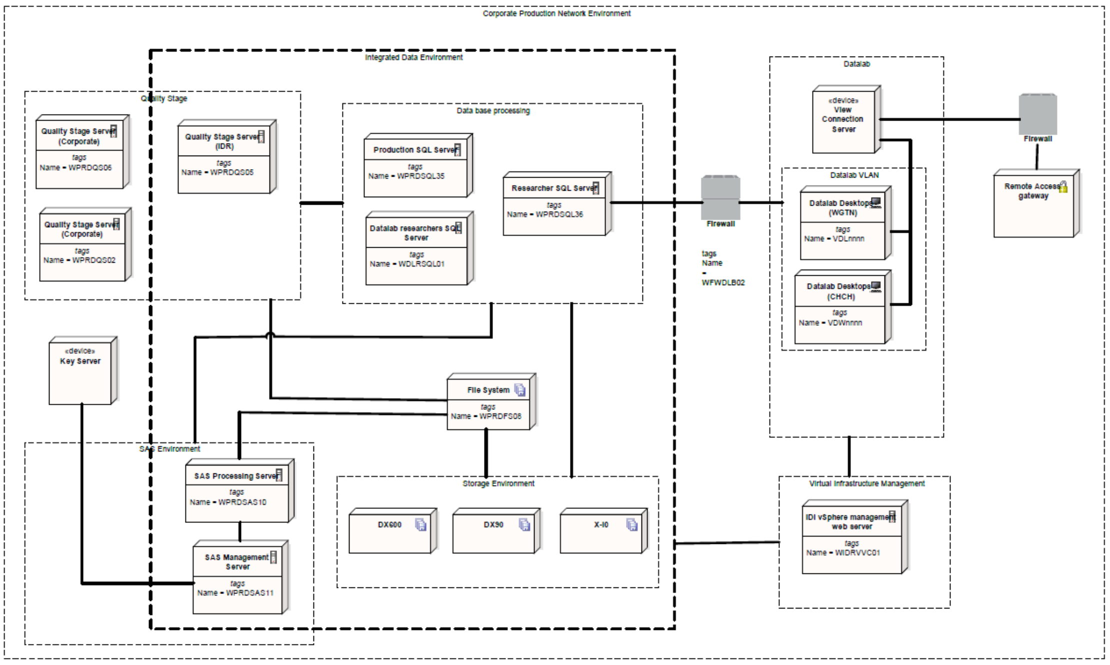
Only external researchers' projects - those whose code begins with the the "MAA" prefix - are included in the analysis in that section. Stats NZ research projects (beginning with the "IMR" prefix) are excluded.

All code is lodged in the Stats NZ document management system (https://stats.cohesion.net.nz/Sites/CR/CRPRS/IDI/ManagementandAdministration/Forms/All%20Documents%20FY.aspx?RootFolder=%2FSites%2FCR%2FCRPRS%2FIDI%2FManagementandAdministration%2FIDI%2Dlayers%2Fhow%2Dis%2DIDI%2Dused&).

Early versions of the report were discussed with the internal Stats NZ advisory group for the "IDI Layers" workstream and with the Census Transformation team. Five individual reviewers generously provided comments. All ommissions and errors remaining are my responsibility.

9 Annex B - IDI architecture

The formal "as-built" architecture of the IDI in the Data Lab is shown below. Note that this excludes the extra SAS server `artsas08` which I understand is a late addition to the Data Lab, and `wprdsql31` which hosts the LBD databases.



10 Annex C - Example normalisation of an IDI table

This is an example script to do a bit more normalising of one of the tables in the IDI and compare sizes, performance, etc. I choose source_ranked_ethnicity because the 6 columns with information in their names really bug me as a bother to match with metadata

`IDI_Clean.data.source_ranked_ethnicity`, the original table used here, is an off-shoot of the Census Transformation project. For each individual snz_uid it has a 1/0 indicator in each of six columns (grp1, grp2, grp3 etc) for whether the best source of ethnic info available for that person has them listed as having that ethnicity. Person-to-ethnicity is a many to many relationship and this is represented in the current data in wide format ie six columns. This makes it harder to safely join with meta data, and also comes at a slight performance cost for the database.

This script is designed to be done interactively with MS SQL Server Management Studio so you can inspect the results as you go.

You need to change the schema name `[DL-IMR2017-05]` to a schema you have permissions to create and destroy tables on (most research projects don't have one; you need to ask access2microdata).

Peter Ellis, 24 August 2017

```
------------------------------------------------
-- First part of script is all about creating the normalised version of the table

-- Drop the normalised version if we're trying this for the second time:
IF OBJECT_ID('IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long', 'U') IS NOT NULL
    DROP TABLE IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long;


-- Gather (ie UNPIVOT) the six ethnicity columns into one long one.
SELECT  snz_uid,
        cast(substring(snz_ethnicity_grp, 18, 1) as int) as snz_ethnicity_grp,
        snz_ethnicity_source_code
INTO IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long
FROM
    (SELECT  *
     FROM IDI_Clean.data.source_ranked_ethnicity) a
UNPIVOT
    (value FOR snz_ethnicity_grp IN
          (snz_ethnicity_grp1_nbr, snz_ethnicity_grp2_nbr,
          snz_ethnicity_grp3_nbr, snz_ethnicity_grp4_nbr,
          snz_ethnicity_grp5_nbr, snz_ethnicity_grp6_nbr)
          ) AS unpvt
WHERE value = 1
ORDER BY snz_uid, snz_ethnicity_grp;


-- Add primary key and a column index.  Need to constrain snz_ethnicity_grp to be not null
-- so it can be part of the primary key:
ALTER TABLE IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long
    ALTER COLUMN snz_ethnicity_grp INTEGER NOT NULL;

-- might *need* to run this bit interactively to pause here before running the next bit...

ALTER TABLE IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long
    ADD PRIMARY KEY (snz_uid, snz_ethnicity_grp);

-- A column index is fancy Microsoft way of making tables for analytics run queries fast:
CREATE COLUMNSTORE INDEX ethnicity_long_column
  ON IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long
    (snz_uid, snz_ethnicity_grp, snz_ethnicity_source_code);
```

Demonstration use of the long, thin version:

```
-------------------------------------
-- Let's compare some query results from the new long version, versus the original.

-- The normalised version is very slightly faster

-- First, from the long version, 0 second:
select count(1) as number, snz_ethnicity_grp
    from IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long
    group by snz_ethnicity_grp

-- Then from the original as in IDI_Clean, 2 seconds a bit longer (and much uglier code):
select
    sum(cast(snz_ethnicity_grp1_nbr as int)) as grp1,
    sum(cast(snz_ethnicity_grp2_nbr as int)) as grp2,
    sum(cast(snz_ethnicity_grp3_nbr as int)) as grp3,
    sum(cast(snz_ethnicity_grp4_nbr as int)) as grp4,
    sum(cast(snz_ethnicity_grp5_nbr as int)) as grp5,
    sum(cast(snz_ethnicity_grp6_nbr as int)) as grp6
from IDI_Clean.data.source_ranked_ethnicity
```

```
-------------------------------------------
-- But the real gain comes from being able to match to metadata


-- The metadata we need is not in the database for some reason so we get it from
-- www.stats.govt.nz/methods/classifications-and-standards/classification-related-stats-standards/ethnicity.aspx

IF OBJECT_ID('IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6', 'U') IS NOT NULL
    DROP TABLE IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6;

CREATE TABLE IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6 (
    snz_ethnicity_grp INT NOT NULL,
    snz_ethnicity_name VARCHAR(50) NOT NULL,
    snz_ethnicity_shortname VARCHAR(5) NOT NULL
    PRIMARY KEY (snz_ethnicity_grp));

INSERT INTO IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6
        (snz_ethnicity_grp,
        snz_ethnicity_name,
        snz_ethnicity_shortname)
    VALUES   (1, 'European', 'Europ'),
             (2, 'Maori', 'Maori'),
             (3, 'Pacific Peoples', 'Pacif'),
             (4, 'Asian', 'Asian'),
             (5, 'Middle Eastern/Latin American/African', 'MELAA'),
             (6, 'Other Ethnicity', 'Other'),
             (9, 'Residual Categories', 'Resid');


-- Now the query, joined with metadata
SELECT snz_ethnicity_name, number FROM
    (SELECT COUNT(1) AS number, snz_ethnicity_grp
        FROM IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long
        GROUP BY snz_ethnicity_grp) a
LEFT JOIN IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6 b
  ON a.snz_ethnicity_grp = b.snz_ethnicity_grp;
```

```
-- A more complex query to produce a tidy dataset of resident population on 30 June of each month
-- by ethnicity (noting that people with multiple ethnicities are double counted). This
-- version is tidy hence suitable for further analysis (eg in ggplot2 or something).
SELECT srp_ref_date, snz_ethnicity_name, number
    FROM
        (SELECT srp_ref_date, snz_ethnicity_grp, count(1) as number
            FROM IDI_Clean.data.snz_res_pop a
            RIGHT JOIN IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long B
            ON a.snz_uid = b.snz_uid
            GROUP BY srp_ref_date, snz_ethnicity_grp) c
        LEFT JOIN IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6 d
            ON c.snz_ethnicity_grp = d.snz_ethnicity_grp
    WHERE srp_ref_date IS NOT NULL
    ORDER BY srp_ref_date DESC

-- or as an untidy for machines but human-friendly cross-tab:
SELECT * from
(SELECT YEAR(srp_ref_date) AS June_30_Year,
        snz_ethnicity_shortname,
        number
    FROM
        (SELECT srp_ref_date, snz_ethnicity_grp, count(1) as number
            FROM IDI_Clean.data.snz_res_pop a
            RIGHT JOIN IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long B
            ON a.snz_uid = b.snz_uid
            GROUP BY srp_ref_date, snz_ethnicity_grp) c
        LEFT JOIN IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6 d
            ON c.snz_ethnicity_grp = d.snz_ethnicity_grp
    WHERE srp_ref_date IS NOT NULL) as SourceTable
    PIVOT (
        AVG(number)
        FOR snz_ethnicity_shortname IN ([Europ], [Maori], [Asian], [Pacif], [MELAA], [Other])
        ) as PivotTable;
-- Note the numbers exceed published official stats a bit, as per paper from Census Transformation.
```

```
--------------------------------------------------------
-- If people want the original wide version we could still have it as a view:

use IDI_Sandpit;

IF OBJECT_ID('IDI_Sandpit.[DL-IMR2017-05].vw_source_ranked_ethnicity', 'V') IS NOT NULL
    DROP VIEW [DL-IMR2017-05].vw_source_ranked_ethnicity;

-- This bit can't be run in batch mode; CREATE VIEW has to be the first command in a batch.
-- So you need to highlight and run this next command as a one-off:
CREATE VIEW [DL-IMR2017-05].vw_source_ranked_ethnicity AS
    SELECT  * FROM
        (SELECT snz_uid, snz_ethnicity_shortname, snz_ethnicity_source_code, 1 as dummy
            FROM IDI_Sandpit.[DL-IMR2017-05].source_ranked_ethnicity_long a
            LEFT JOIN IDI_Sandpit.[DL-IMR2017-05].dim_snz_ethnic_6 b
                ON a.snz_ethnicity_grp = b.snz_ethnicity_grp) as SrcTab
         PIVOT (
            AVG(dummy)
            FOR snz_ethnicity_shortname IN ([Europ], [Maori], [Asian], [Pacif], [MELAA], [Other])
                ) as PivotTable;

-- With SQL Server you can also index a view to make it run faster but it gets a bit complicated
-- so I won't bother for now.


/*
So this was an interesting exercise.  What does it tell us about the data model and about layers?

* The current wide format doesn't lend itself to making views or combining things with metadata,
  but it's not too hard to fix.  The question is when and where...
* Some metadata isn't there yet (unless I've looked in the wrong spot)
* Changing the overall data model might need to be done before we do any creating of views and tables
  and stuff to make up an "analytical layer".

*/
```

11 Annex D - Speeding up basic queries in the LBD with indexes

The code below generates around a 100x speed up of some basic queries on one of the tables in the LBD

```
/*Demonstration of how much speed up might be possible with some indexing in the LBD*/

-- Some unacceptably slow queries:
-- 31 seconds:
SELECT TOP 1000 *
  FROM [ibuldd_clean].[dbo].[fact_aes_enterprise_year]
  WHERE dim_year_key = 201503;

-- 21 seconds:
SELECT DISTINCT(DIM_YEAR_KEY) FROM IBULDD_CLEAN.DBO.fact_aes_enterprise_year;

-- 36 seconds:
SELECT COUNT(1) FROM IBULDD_CLEAN.DBO.fact_aes_enterprise_year;

--Copy the table into a temporary table called #aes that I can create indexes on
SELECT *
    INTO #aes
    FROM IBULDD_CLEAN.DBO.fact_aes_enterprise_year;

-- Check performance is still slow; 14 seconds:
SELECT DISTINCT(DIM_YEAR_KEY) FROM #aes;

ALTER TABLE #aes ADD PRIMARY KEY (enterprise_nbr, dim_year_key);
CREATE INDEX YR ON #aes (dim_year_key);


-- 0 seconds:
SELECT DISTINCT(DIM_YEAR_KEY) FROM #aes;

-- 0 seconds:
SELECT count(1) FROM #aes;

-- 0 seconds
SELECT TOP 1000 *
  FROM #aes
  WHERE dim_year_key = 201503

DROP TABLE #aes
```

12 Annex E - Researcher project folder case studies

As part of this exercise, a dozen research project folders were examined reasonably closely. Findings from that examination informed the discussion in the main body of the report. The notes on four of a range of approaches are provided below as examples.

12.1 Letter to researchers obtaining permission to examine research folders

From: Access2Microdata - Shared Mailbox access2microdata@stats.govt.nz (mailto:access2microdata@stats.govt.nz)

Sent: Monday, July 24, 2017 2:04:03 PM

To: Access2Microdata - Shared Mailbox

Subject: StatsNZ: Permission for Stats NZ to access IDI research projects' folders to help in designing information and analytical layers

Dear lead researchers

Some of you will be aware that Stats NZ is commencing a range of work to improve the usefulness and usability of the Integrated Data Infrastructure (IDI). As part of this overall programme of work, we have contracted Peter Ellis to identify options for "information and analytical layers" in the IDI, to reduce the burden on data exploration for current users and enable the development of simpler access tools for novice data users. Peter's analysis will build on prior work from some of you and the on-going discussions in the community.

As part of this exercise, Peter has identified the need for a better understanding of how researchers currently use the IDI, to best appreciate the opportunities for reducing researcher burden. Building this understanding can draw in part on interviews and meetings with some of you, but to be really concrete it should also involve examination of the actual analysis – that is, people's code and working folders in the IDI Data Lab. I am writing to you to seek your permission for Peter to include your research project in his analysis. This will mean giving him access rights to your research folders so he can identify patterns, good practice, and apparent roadblocks and frustrations from how people set up their folders, perform analysis, document their approaches for themselves and other researchers, etc.

If you have any concerns could you please let me know by close of Monday 31 July 2017. Peter's time with us is relatively short and we're keen to make the most of it to improve our service offerings for you. If we do not hear from you by then we will assume you have no problem with Peter undertaking this work. No action is needed on your part if you are comfortable with this approach, it can all be done centrally from our own building – you certainly do not need to tidy up your folders (and in fact we strongly discourage this, so Peter can get the best sense of how analysis is done on the coal face).

We appreciate this could appear a little intrusive! So Peter will be bound by research and evaluation ethical standards in this work, as well as standard Data Lab access provisions. No individual projects or researchers will be identified in his analysis or report, or in discussion by him with Stats NZ or anyone else. All intellectual property associated with your project will remain as under any previous arrangements.

Many of you will know Peter and his experience with both data analysis and evaluation of sensitive projects and programmes; he is the former Lead or General Manager Evidence and Insights at the Social Investment Unit/Agency, Manager Sector Trends at MBIE, and Director Program Evaluation for AusAID.

Thank you for your consideration in this matter.

Regards,

Andrea

Andrea Blackburn

Senior Manager Integrated Data | Kaiwhakahaere Hōtuku Kōmitimiti

Stats NZ Tatauranga Aotearoa

12.2 Case A

12.2.1 Overall summary

Relatively small focused project. Single analyst in the Data Lab, single purpose, simple cross tabs output

12.2.2 Data

Three schemas in a date-stamped version of `IDI_Clean`, including `data`.

12.2.3 Folders

- No sub folders other than for output checking.

12.2.4 Approach

- SAS used for data extraction, management and analysis; Excel for random rounding and release
- SAS scripts are numbered in the order they need to be executed
- SQL embedded in the SAS scripts
- Analysis was clearly exploratory and iterative, building the data into more complex forms a bit at a time
- Some `.sas` files from other projects in the folder but not clear that these were particularly helpful
- A number of `.sas7bdat` files which are created by the various `.sas` programs and then built on; data extractions from `IDI_Clean` are slow enough to want to start from the intermediate set.
- Absolute file paths hard coded
- Analysis mostly reproducible; a small number of dependencies (reference data) appeared to have been sourced externally and dropped in
- The intermediate data assets created by this project quite specific to its particular cohort and only of use for people studying the same thing
- code is annotated and has headers explaining general purpose but not much detail.

12.3 Case B

12.3.1 Overall summary

Complex large project, with extensive data preparation challenges, sophisticated statistical modelling, and a big range of output.

12.3.2 Data

Nearly everything in the IDI.

12.3.3 Folders

This project is technically a sub-project in a much wider programme of work and part of a complex folder system. Individual analysts (at least ten on this project) have named folders, and there is a folder specifically for the sub-project.

12.3.4 Approach

- SAS used for data preparation, R for statistical modelling, then back to SAS.
- Intermediate datasets stored in `IDI_Sandpit`
- uses SIAL and SIA's "Data Foundation" macros, so reproducibility depends on first having those working in the same versioning
- code is carefully curated, follows style guide, and explicitly belongs to the larger team not to individuals
- master scripts running secondary scripts; extensive use of SAS macros (custom for this project and the multi-purpose SIA "Data Foundation" macros")
- documentation and comments aimed at people picking up the code to repeat it rather than to developers / maintainers
- RStudio project for the statistical modelling stage, but still has absolute file paths in both the R and SAS code
- The intermediate data assets created by this project quite specific to its particular cohort and only of use for people studying the same thing

12.4 Case C

12.4.1 Overall summary

Wide range of research and analysis - a classic mega-programme of analysis. Analysis ranges from lots of simple cross tabs, to traditional research and evaluation, through to simulations to support policy develop0ment.

12.4.2 Data

Nearly everything in the IDI. Researchers work for a Ministry and have a particular interest in the most relevant schemas, but routinely are linking. Interested in causality in both directions ie other issues impacts on their portfolio and vice versa.

12.4.3 Folders

Complex folder structure. Individual researchers have their own folders for exploratory work and some development, but most projects take place in folder systems for that project. Lots of duplication of macros and snippets of SAS code and evidence of a version control nightmare, but a centralised location for the standard macros and libnames

12.4.4 Approach

Mostly in SAS all the way through, with a relatively small amount of dedicated SQL files. SQS extensively uses `PROC SQL` followed by SAS-native `DATA` steps for additional manipulation in the language researchers are more comfortable with. Coding style is varied - some extremely well structured and documented code, some not so much; evidence of variance at the individual level.

Repeat tasks have been abstracted into parameterised macros, particularly when dealing with defining indicator variables and similar for populations.

12.5 Case D

12.5.1 Overall summary

Econometric project focused on a single research question.

12.5.2 Data

Individual economic data in the IDI.

12.5.3 Folders

Signs of a small number of researchers, most of the work by a single author. Project is not portable because of absolute file paths.

12.5.4 Approach

Well structured `.sql` files that all run smoothly and were apparently used to export CSV files by hand. Well ordered Stata files that produce logs and output, which includes both exploratory cross tabs and a range of regression results.

13 Annex F - Approaches to data warehousing

As mentioned in the background, a data warehouse is a central repository of integrated data from multiple sources. It includes:

- a staging area and integration layer
- the warehouse or vault itself for holding the data
- datamarts for presenting data to users.

The various stages of data in its lifecycle are moved through the warehouse system via computer programs referred to as "ETLs" (Extract-Transform-Load).

There is controversy over how data warehouses should be designed and built, with two main schools of thought:

13.1 Bottom up or “Kimball” method

Named after Ralph Kimball, the inventor of dimensional modelling. The essence of the Kimball method is to build datamarts for the different sets of users, carefully ensure that standards for shared dimensions are agreed upon and used in common, and declare the “warehouse” to be the sum total of the datamarts in the system. The data is not necessarily ever stored in fully normalised form, but in a partially normalised star schema that is designed to match how analysts think of the data generation process and is reasonably efficient for queries (eg avoids too many joins).

13.2 Top down or “Inmon” method

Named after Bill Inmon, this more comprehensively planned and designed method involves first designing a fully normalised data model for all data to be captured in the warehouse. All data is stored in the warehouse in this fully normalised form, but datamarts present the data in whatever data model is appropriate for the analytical purposes of the datamart.

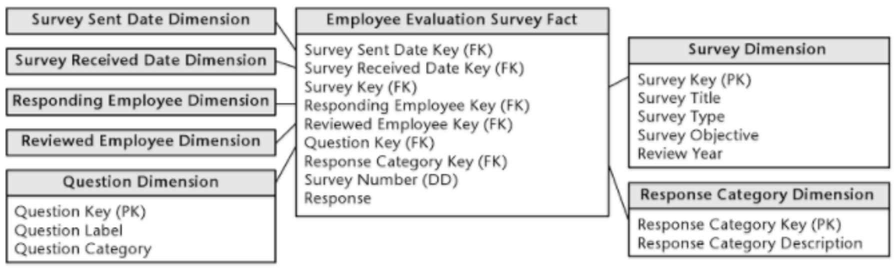
13.3 Discussion

It can be seen that for the end user, the result may appear similar - either way, they get a datamart that is presented in the form suitable for their needs, which might be a partially normalised star schema even under Inmon’s method. The big difference in the end proudct is under the hood, and there is a sequencing difference on what is to be done first when embarking on a warehousing project. Many parties insist “there is no right or wrong” between the two ideas but a choice needs to be made on the basis of how projects and change is to be managed, how soon results are needed (Kimball’s method can obviously start delivering datamarts earlier but Inmon’s method, once delivered is a more solid basis for delivering in the time frame beyond one year), etc.

The IDI and LBD as currently constructed do not fit with either of the methods.

- Key parts of the data eg full text of survey questions are either not in the datamart at all or in inconvenient form
- The data does not go at any stage into a fully normalised data model as per Inmon’s method but exists only in the partially normalised form seen by researchers
- As discussed in the main body of the report, neither the IDI or LBD is presented in a pure Kimball dimensional model, although they both echo elements of that approach and the LBD in particular goes part way and has tables named as facts and dimensions

Given that there is no “third normal form” version of the data but there is agreement that data can be presented as only partially normalised for analysts’ ease of understanding and use, one question could be “how partially normalised should that be”? There are valid alternative views on this, but the answer is definitely “more normalised than is currently the case for the IDI and LBD”. This can be seen most clearly with the case of surveys. Kimball’s book *The Data Warehouse Toolkit* (https://books.google.co.nz/books?id=XoS2oy1IcB4C&pg=PA197&lpg=PA197&dq=kimball+human+resources+survey+schema&source=bl&ots=1CEdmylLaD&sig=mXZScbbuAWt7nIvY7Xfc-pu18X4&hl=en&sa=X&ved=0ahUKEwju4oHBkYXWAhUDVrwKHYbGCYQQ6AEIJzAA#v=onepage&q&f=false) includes a sample dimensional model for warehousing multiple surveys with differing purposes and question instruments, which is shown below (in his case, the survey is of employees - substitute “respondent” for “employee”).


Survey Sent Date Dimension
Survey Received Date Dimension
Responding Employee Dimension
Reviewed Employee Dimension
Question Dimension
Question Key (PK)
Question Label
Question Category
Employee Evaluation Survey Fact
Survey Sent Date Key (FK)
Survey Received Date Key (FK)
Survey Key (FK)
Responding Employee Key (FK)
Reviewed Employee Key (FK)
Question Key (FK)
Response Category Key (FK)
Survey Number (DD)
Response
Survey Dimension
Survey Key (PK)
Survey Title
Survey Type
Survey Objective
Review Year
Response Category Dimension
Response Category Key (PK)
Response Category Description
Figure 8.7 HR survey schema.

Note that elements that in the IDI and LBD are regarded as “metadata” (and either stored in a separate database, or only available in non-database collateral) such as the full text of questions are in fact simply dimension tables in the data model.

The key difference of this approach to the wide table data models for surveys in the IDI and LBD is that instead of there being a column for each survey question, there is a row for each question - respondent combination, with generic `Question Key` and `Answer Key` columns. The data could be pivoted back to wide format as a view if that’s what analysts wanted, while still getting the benefits of the stable data model for the underlying tables.

Kimball’s recommended approach abstracts away from the specific questions in each survey, so if the questions change the data model of the datamart does not need to change but it just gets new values in the `Question Dimension` and `Response Category Dimension` tables. The suggested schema is too simple for surveys in the IDI and LBD (in which category I include the Census and survey-like Inland Revenue forms) and would need to be adjusted with hierarchies of questions and different response categories, but the principle could be readily used, as discussed in the main body of the report in the context of the US Agriculture Department example.