April 2022

IDI exemplar project

Guidance and training

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

TE ROUROU TĀTARITANGA
Informatics for Social Services and Wellbeing

New Zealand Government

Author

Simon Anastasiadis

Acknowledgements

Data Services Team, Stats NZ, without whom we would not have the Integrated Data Infrastructure and whose ongoing dedication sustains and grows it.

Te Rourou Tātaritanga, Informatics for Social Services and Wellbeing (terourou.org/)

Creative Commons Licence

Liability

Citation

Social Wellbeing Agency 2022. IDI exemplar project – Guidance and training. Wellington, New Zealand.

ISBN 978-0-473-62595-5 (online)

Published in April 2022 by
Social Wellbeing Agency
Wellington, New Zealand

Contents

Introductory end-to-end IDI exemplar project

Integrated data can be a powerful tool for research. However, the integration of different data sources does not guarantee these sources are all straightforward to analyse. New Zealand's Integrated Data Infrastructure (IDI) and Longitudinal Business Database (LBD) can be intimidating environments to work in for unfamiliar researchers.

This exemplar has been written to provide new researchers to the IDI with a simple end-to-end project. It is focused on the practical aspects of managing a project and manipulating the data, after research questions and goals have been agreed. The project reflects our current best practice, and we hope that it provides a useful guide for researchers to learn from. You can download the exemplar from our GitHub page.[1]

This exemplar has been written using SQL and R code. Following the exemplar in detail requires familiarity with at least one of these languages (or at least a willingness to look up key terms). Researchers who use other programming languages will still benefit from following this exemplar. The structure of our approach is not specific to any language or to the implementation in this exemplar. Researchers may wish to write equivalent code in their preferred programming language.

In the course of our research in the data lab, we have developed a collection of tools and scripts to improve the quality and speed of our work. Several of these have been integrated into this exemplar. Researchers are free to reuse these resources for their own work. But, like the choice of language, our overall approach is not specific to these tools.

Three challenges to delivering analytics

Our motivation for this exemplar and the best practice it reflects has been to improve the effectiveness with which we deliver analytic projects. As part of developing our best practice we have had to respond to three challenges to delivering research using integrated data:

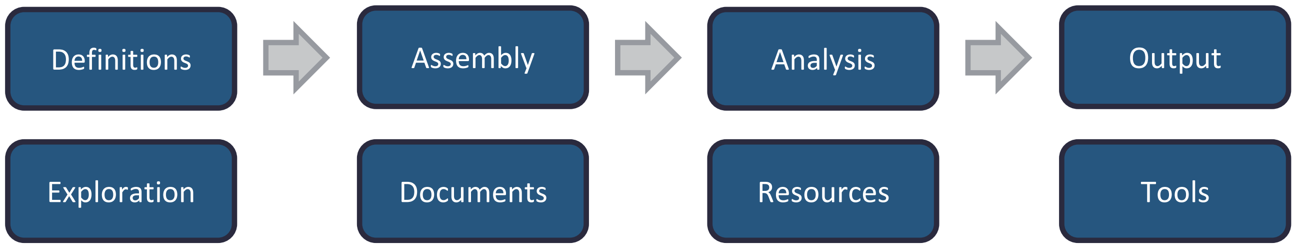
1. The data available covers a wide range of domains and we do not have consistent access to subject matter experts for every domain or dataset.
2. Customers' desire for how soon results will be delivered.
3. The messy and complex nature of IDI research and differing approaches to handing the work can lead to missed opportunities for collaboration, efficient onboarding of new staff, and reuse of previous work.

Creating structure to overcome challenges

In response to these challenges, we have found it effective to structure each project into the same set of tasks. The figure below gives our recommended structure.

[1] https://github.com/nz-social-wellbeing-agency/idi_exemplar_project

Figure: Overview of recommended structure



This structure provides several advantages to our staff that improve the effectiveness of our delivery and supports us to overcome the challenges identified above.

- By dividing the project into component tasks, we reduce the complexity staff must handle at any one time. This simplifies problem solving, reduces errors, and provides clear steps for new researchers learning the project.
- Data processing in this structure is linear. By separating out the tasks that deliver the project outputs (top row), we make it clear how inputs are transformed to outputs. This makes it easier to review or audit the work, and it is clear where to focus if the project is rerun.
- By recording definitions and exploration, subject matter knowledge developed in each project is preserved and becomes a resource to accelerate future work. Having these files easy to find in their own folder increases their reuse in future work.

In addition, the boundaries between the tasks have been chosen to enable collaboration. For example: several researchers can create definitions or conduct exploration in parallel, the staff managing the assembly can be different from the staff conducting the analysis, and tools developed by one staff member may be used by another to produce output.

Follow the exemplar through a basic analysis

This exemplar starts from an empty project, works through data processing, and concludes with an output submission. The information this project produces from the IDI is age distributions and average income measures for each region.

The folders and files provided in the exemplar project are a close approximation to the final state of the project. We have removed temporary files and any data that would pose a confidentiality concern from the published exemplar. Where files have been removed these are noted in the exemplar. A complete copy of the exemplar including the sensitive files should be available within the data lab on the data lab wiki.

Where we refer to specific folders or files in this section, these are given in **bold**. To keep this document short, we will describe the purpose of specific files and folders but not their workings in detail. Readers wanting to understand the full details will need to review the contents of each file.

Some files will require editing before they can be run. This will often just be file paths, and project codes (e.g. MAA20YY-XX) because each researcher is limited to using only their projects and folders.

In some cases, there will be multiple versions of a file. In these cases, the files differ by approach (for example, choice of programming language or tool used) and the suffix of the file indicates the approach taken. We will only refer to the core of the file name (omitting the suffix) as the purpose of the file, not the details of its implementation, are important for this guide.

Tip: As most IDI researchers are familiar with SQL, we recommend that the first time through this exemplar you use only the SQL files. This will simplify the initial learning. As your project grows more complex, then it is worth considering using a wider range of programming techniques.

Structure the project around key tasks

Many data lab research projects involve the same series of tasks. Keeping these tasks separate helps ensure the process is clear, making collaboration or revisions more straightforward. Arranging a project with subfolders for each task is an effective way to keep the key tasks separate.

The exemplar project uses a structure common to many of our projects. We provide an overview of the purpose of each folder along with its name:

- **_For checking** – Contains data lab output that has been submitted to Stats NZ for checking or is awaiting checking. This makes it easy for Stats NZ staff to find our output.
- **_Checked** – Contains data lab output that has been reviewed by Stats NZ checkers. This is useful for us finding previous outputs and supports the Stats NZ checking process.
- **Documentation** – Contains project plans, working notes, and supporting information.
- **Data Exploration** – Contains code and notes from investigations assessing the quality or suitability of data or techniques.
- **Definitions** – Contains definitions of the study population, variables, and measures that have been prepared for use in the research dataset.
- **Assembly** – Contains the code to combine the individual population and measure definitions into a single research-ready dataset.
- **Analysis** – Contains the code to clean the dataset, analyse the data, and output the results.
- **Output** – Contains summarised tables and research results. These files are often produced by code and are then prepared for checking and copied to **_For checking**.
- **Resources** – Contains assorted code and templates that are useful across projects.
- **Tools** – Contains formalised tools that are useful across projects.

A folder structure in this pattern, but without any of the exemplar files, can be found in the zip file **empty folder structure** in the **Resources** folder.

Tip: As you undertake data lab research, build a collection of reusable definitions, resources, and tools to make future research easier.

Setup and test the tools

Some tools require configuring prior to use. This is most likely at the start of a project when file paths and project codes are different from the last use of the tool.

The exemplar includes the Dataset Assembly Tool in the **Tools** folder, for ease of creating the analysis-ready, research dataset. Setup for this tool requires entering connection details in **dbplyr_helper_functions**. It is also recommended that you run **automated_tests** to confirm the tool is running correctly. Full instructions for setting up this tool can be found in the Dataset Assembly Tool training presentation online[2] and in the **documentation** subfolder of the tool. The code for the tool is available on both the Data Lab Wiki and our GitHub page[3].

You can skip this step if you are not intending to use any of the tools. For example, you are focusing on only the SQL files for your first time through the exemplar.

Tip: When creating and sharing tools, save test scripts with each tool. This simplifies the process of testing whether the tool is performing correctly.

Explore the data to build understanding

Even when there is detailed data documentation, practical experience with a data table is important for staff to use it with confidence. Given the breadth of information in the IDI, researchers will likely need to build familiarity with new datasets during their projects.

For the exemplar, two investigations can be found in the **Data exploration** folder: First, we have investigated the income tables. There are four different income tables, and we want to be confident of their consistency before proceeding. Our exploration and notes appear in **assess_different_income_tables**.

Second, we investigate which column produces the best join of metadata to data in **test_meshblock_year_codes**. Because there are multiple columns of meshblock codes and the documentation does not specify which year to use, we test several options and choose the best one when we write our address definition.

We tend to store all the work that is not part of producing our final output in this folder. While some of this work could be stored elsewhere (such as in the analysis folder), storing this work here makes it very clear what work is part of producing the key output and what work is not.

When deciding whether a code file belongs in this folder, we find it useful to ask: “Would a new staff member to this project need to know about this file?” and “If I have to come back to this project in six months (to modify or rerun it) would I need to know about this file?” If the answer is

[2] https://swa.govt.nz/assets/Publications/guidance/Dataset-Assembly-Tool-introduction-and-training-presentation.pdf

[3] https://github.com/nz-social-wellbeing-agency/dataset_assembly_tool

"no" then the file probably belongs in the exploration folder. If the answer is "yes" then the file probably belongs elsewhere.

Depending on your research style, you may find this folder becomes very full and requires subfolders. Structure these in a way that makes most sense to you and your project team.

Tip: Save the learnings and notes from data exploration alongside the exploration code. This reduces the chance of repeating previous investigations.

Define the study concepts prior to use

For our project to output age distributions and average income measures for each region, we will need to know who lives in New Zealand, which region they live in, what they earned, and when they were born.

Rather than create all of these within the same piece of code, we define each concept by itself and then assemble the different definitions into a single research dataset. This way each definition is independent until they are assembled together, and we can avoid code tangles where a change in one file can lead to a cascade of changes through dozens of files.

The **Definitions** folder contains the definitions needed for this analysis:

- **residential_population_2020** produces a table containing the 2020 residential population.
- **annual_taxable_income** produces a table from which we can read total taxable income.
- **address_descriptors** produces a table containing address information for mid-2020 including region.

Each file outputs either a View to IDI_UserCode or a Table to IDI_Sandpit. The IDI_Sandpit and IDI_UserCode are two databases where researchers can write data that they are preparing or processing. By writing our definitions to these database locations, we create permanent objects that can be reused later on.

Tips:

- When naming tables prefixes can be used to group similar tables together (e.g. "defn_" for definitions) and suffixes can be used to indicate versions.
- Avoid creating Views of Views – it hides the complexity and adds inefficiency.
- Best practice when creating Tables is to add non-clustered indexes and to compact large tables. Indexing is demonstrated in **address_descriptors.sql**. See the "Care for the shared environment" section below for more details on compacting.

A note on the difference between Tables and Views

While both are informally referred to as tables or data, Tables and Views are very different ways of storing information in the database. A Table is the standard way data is stored in a database – data is written to disk so it can be requested later. A View is a virtual table that is constructed each time it is requested. This means that a View takes very little hard drive space to store but is slower to read than the equivalent table.

We recommend using Views where preparation is simple – renaming and filtering existing data, at most one join, no nested queries – and using Tables for information that requires more complex preparation. This is an efficient trade-off between use of hard drive space and performance.

Assemble definitions into single table

Once definitions for the research have been constructed these can be assembled together into a single table ready for analysis. We do this using the **run_assembly** code in the **Assembly** folder.

When doing assembly without supporting tools, we must write code that combines the definitions. This is simple for small projects but can become impractical as the number of inputs increases.

When doing this with the Dataset Assembly Tool, we provide two control files **measures** and **population_and_period** that contain the instructions for the assembly. When doing assembly with the tool, we must first setup the tool according to its instructions[4].

Regardless of the approach taken, the assembly instructions will likely be a useful resource during the rest of the analysis. Because the assembly links the input definitions and the research table, we can refer back to it any time we need to check the origin of our data.

Tip: Assemble research datasets in as few steps as possible. This makes the process clearer and easier to update and rerun.

Tidy and analyse the dataset

Now that a research dataset has been created, we are ready to conduct our analysis. As part of this we also tidy the dataset to ensure all the variables are in their preferred format.

The **tidy_variables** script in the **Analysis** folder computes age from data of birth, constructs age categories, handles missing values for income, and collapses multiple indicator columns into one.

To generate an overview of the dataset we can use the *explore* package in R to produce a report on all the variables as part of **tidy_variables**. Note that you may need to install the explore package manually. Instructions for how to install an R package from file can be found in the **Resources** folder.

Because most of the value of research is generated during this step, we have designed our process, and this exemplar, to get to this stage as efficiently as possible.

[4] https://swa.govt.nz/assets/Publications/guidance/Dataset-Assembly-Tool-introduction-and-training-presentation.pdf

Summarise results for release

All results removed from the data lab must be checked by Stats NZ staff to ensure they protect the privacy and confidentiality of people whose data has been used. The rules for this protection are given in the Microdata Output Guide.

The vast majority of data lab output is summary tables containing counts and totals. This is all we need for this exemplar. For creating only these types of outputs the key rules to focus on are:

- Raw counts of individuals less than 6 must be suppressed.
- Totals created from the sum of less than 20 individuals must be suppressed.
- Counts must be randomly rounded to base 3 (RR3).

To produce these results, we first summarise the data using **output_results** from the **Analysis** folder. This produces two csv files in the **Output** folder: **age summary** and **income summary**.

If we output results using our R tools, then we also have the option to confidentialise them using a related tool. Our summarise and confidentialise tools are included together in the exemplar in the same folder as the assembly tool. Documentation for these tools is available here[5] with our published guidance.

Tip: Even though we want to report mean income, we output total income and count of people with income. This is because totals and counts let us calculate means, and it is faster to show we have followed the rules for these outputs than the rules for means.

Confidentialise results and submit to checking

To prepare our output submission, we begin by copying **output template** from **Resources** to create the file **summaries by region** in the **_For Checking** folder. Into this file we copy the results from the **Output** folder and proceed to confidentialise them.

When confidentialising files, rather than overwrite the raw data, we produce the confidentialised copy of the data beside the raw data. This makes it easy to validate that the rules have been applied correctly.

In **Resources** you will also find two macros by Stats NZ to help with confidentialisation: **random_round_rr3** and **RR3 Checking Macro**. The first will apply RR3 on highlighted cells in Excel. The second lets us check our own output. To use them you will need to open both the macro and your output file. More detailed instructions for running each macro are found in the file.

Finally, before we submit, we make two copies of the file. We label one **summaries by region_RAW NOT FOR RELEASE**. This file will be used by Stats NZ checkers to verify we have applied confidentiality correctly. The other file remains named **summaries by region**. We delete all

[5] https://swa.govt.nz/assets/Publications/guidance/summarise-and-confidentialise-tools-training-guide-v2.pdf

the raw data from this file, leaving just the confidentialised data and submit this file to Stats NZ for release.

If we are being thorough, in addition to the RR3 checking macro, we check our own output using checking tools prior to submission. Like the summarise and confidentialise tools, a copy of these is available with the assembly tool. Instructions on the use of the checking tools is available here[6].

Tip: Be nice to your checker! Before submitting your output file, review the file as if you were the checker (or have a colleague review it). If it is hard for you to review the file, then it will be even harder for the checkers.

Care for the shared environment

The data lab is a shared environment, so resources like storage space and computing power are shared between researchers. Respecting the environment, and Stats NZ's provision of it, is an important part of being a good data lab citizen.

There are three key ways we do this:

- During a project, we use efficient data processing to ensure that memory and compute are available for other researchers.
- After key stages in the project, we delete unused tables to free up memory. These could be Tables in IDI_Sandpit, Views in IDI_UserCode, or data files in the project folder. This is safe to do because we can recreate these tables should we ever need them again, by rerunning the project code.
- Submit code and note files for checking so we have these saved outside the data lab. This ensures we are not dependent on Stats NZ storing our files should we wish to reuse them. It also makes it easier to share and collaborate with other researchers.

As part of this exemplar, we wrote several definitions tables and research tables to the database: This is a good time to go and remove them.[7] The **Get_SQL_table_sizes** and **Get_SQL_views** scripts in the **Resources** folder support this by listing all the researcher created tables that a user can access. They also contain code that can be used to delete Tables and Views that are no longer required. Our code may also have generated temporary files: These can all be deleted too.

In addition to listing the researcher created Tables, **Get_SQL_table_sizes** will also return the size of each Tables. This is an effective way to identify large tables that could be compacted to save space. There is example code in **Get_SQL_table_sizes** for compacting a Table.

[6] https://swa.govt.nz/assets/Publications/guidance/self-checking-tools-training-guide.pdf

[7] In IDI_Sandpit look for defn_address_descriptors, exemplar_assembled_data, exemplar_rectangular, and exemplar_tidy. In IDI_UserCode look for defn_residents and defn_annual_taxable_income.

Tip: Code and notes are subject to the same confidentiality rules as other outputs. An effective way to ensure this is to write notes and comments as general statements (“of the 12,000 people, less than 0.1% are affected”) rather than specific statements (“of the 12,345 people, 7 were affected”).

Building on this exemplar for further analysis

The IDI can be challenging to work with for researchers who are unfamiliar with it. We hope this exemplar project provides a template that assists you carrying out your own projects.

One way to use this exemplar is to progressively adapt each part of it so that it is in line with your intended research. This could be done as follows:

- Change the study population – the list of individual identities included in the analysis.
- Change the study time period – the date range within which we are interested.
- Copy an existing definition from another researcher and add it to the project. (you can find some of the Social Wellbeing Agency’s definitions on our GitHub page[8]).
- Create a new definition for a measure that needs to be added to your analysis.
- Add the new definitions into your assembly code and rerun the dataset assembly.
- Revise your analysis, summarising additional variables or producing alternative summaries of the same variables.
- Confidentialise and output your new results.

Even if you do not intend to adapt this exemplar, we recommend that you consider each of the steps above and identify where in the exemplar these changes would be made. This will test your understanding and reinforce any learning so you can continue to develop as a researcher.

Combining this approach with existing practices

Few researchers begin working in the data lab without previous analytic experience. With previous experience comes existing preferences for how to arrange and undertake analytic projects. These preferences may differ from the approach recommended in the exemplar. Even very new researchers will likely belong to teams with established practices and these practices may not be consistent with the approach recommended here.

As a first step, we recommend you adopt the folder structure from the exemplar. Even if you make no other changes to how you carry out your research, storing your project work in this structure will help keep the different stages of the process separate and improve the clarity of your work.

[8] https://github.com/nz-social-wellbeing-agency/definitions_library

Adapt these ideas for use outside the data lab

This exemplar has been written to guide researchers working with integrated data in the data lab. While some parts of this guide are specific to the data lab environment (such as the confidentiality and output process), many of the underlying ideas are applicable to analytic projects in general.

If adapting the ideas in this exemplar outside the data lab, the process to create your research ready dataset is likely to look different. For desktop-based projects where all the data came in spreadsheets or csv files, we recommend replacing the **Definitions** folder with a **Raw data** folder. The **Assembly** folder will then contain the code to produce a research ready dataset from the raw input data.

For a project drawing data from a corporate database, the **Definitions** and **Assembly** folders might be combined into a single **Data preparation** folder. This folder would still contain the code to produce a research ready dataset. But this might be completed in a single stage where the corporate database is well maintained.