Dataset Assembly Tool

Intro and training

December 2020

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

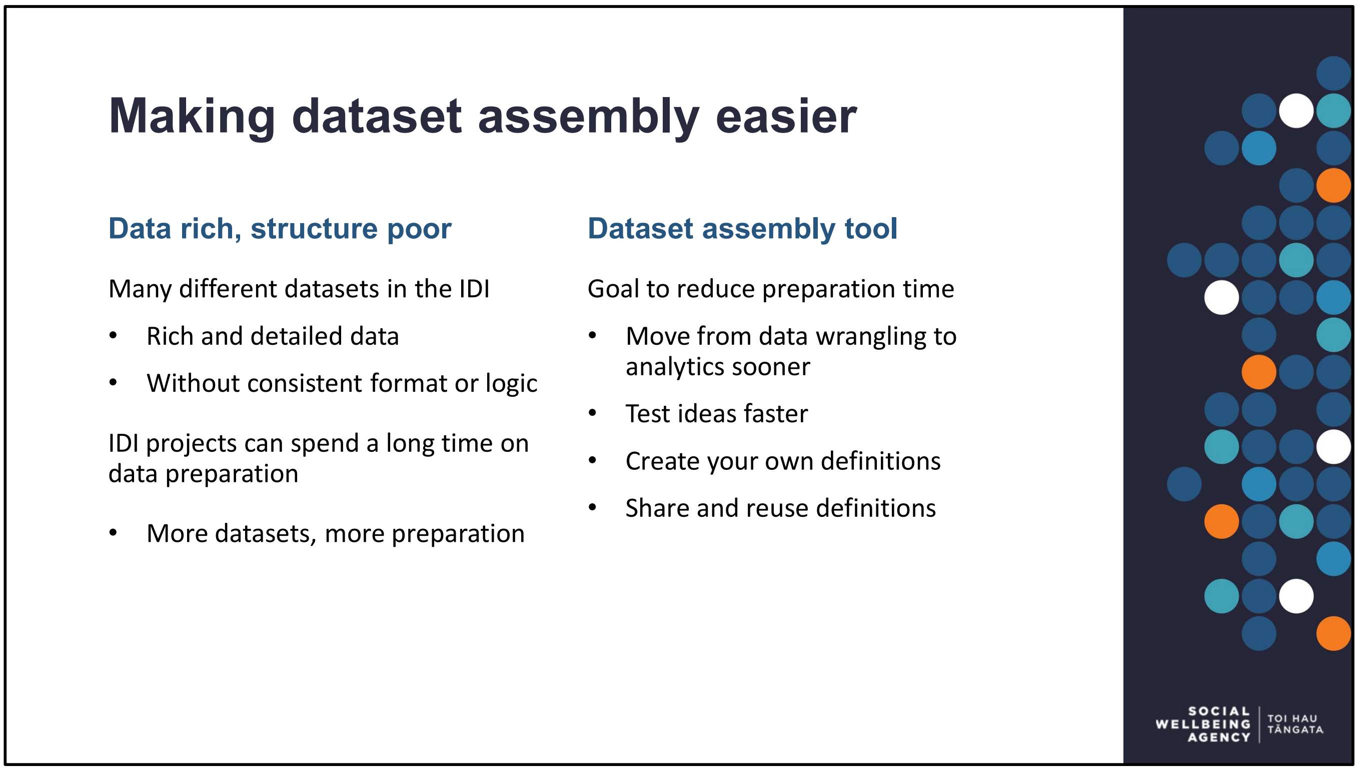
Author: Simon Anastasiadis

The Dataset Assembly Tool is a resource to make data wrangling faster and easier, to improve the quality and timeliness of IDI research.

This presentation includes additional details in the speaker's notes – reading through the presentation is also recommended.
You may want to print the presentation as "Notes Pages", with the slides and the presenter notes.

This presentation should be read/watched alongside "Accelerating Dataset Assembly: Primer and guide to the Dataset Assembly Tool", which discusses the underlying approach and provides further technical information as well as a worked example.

We include images of code. ACTUAL CODE IN THE TOOL MAY HAVE BEEN UPDATED SINCE THIS PRESENTATION WAS MADE. So notes refer to the code in the images in this presentation, users will have to use their best judgement when comparing this presentation to the working tool.



Making dataset assembly easier

Data rich, structure poor

Many different datasets in the IDI

- Rich and detailed data
- Without consistent format or logic

IDI projects can spend a long time on data preparation

- More datasets, more preparation

Dataset assembly tool

Goal to reduce preparation time

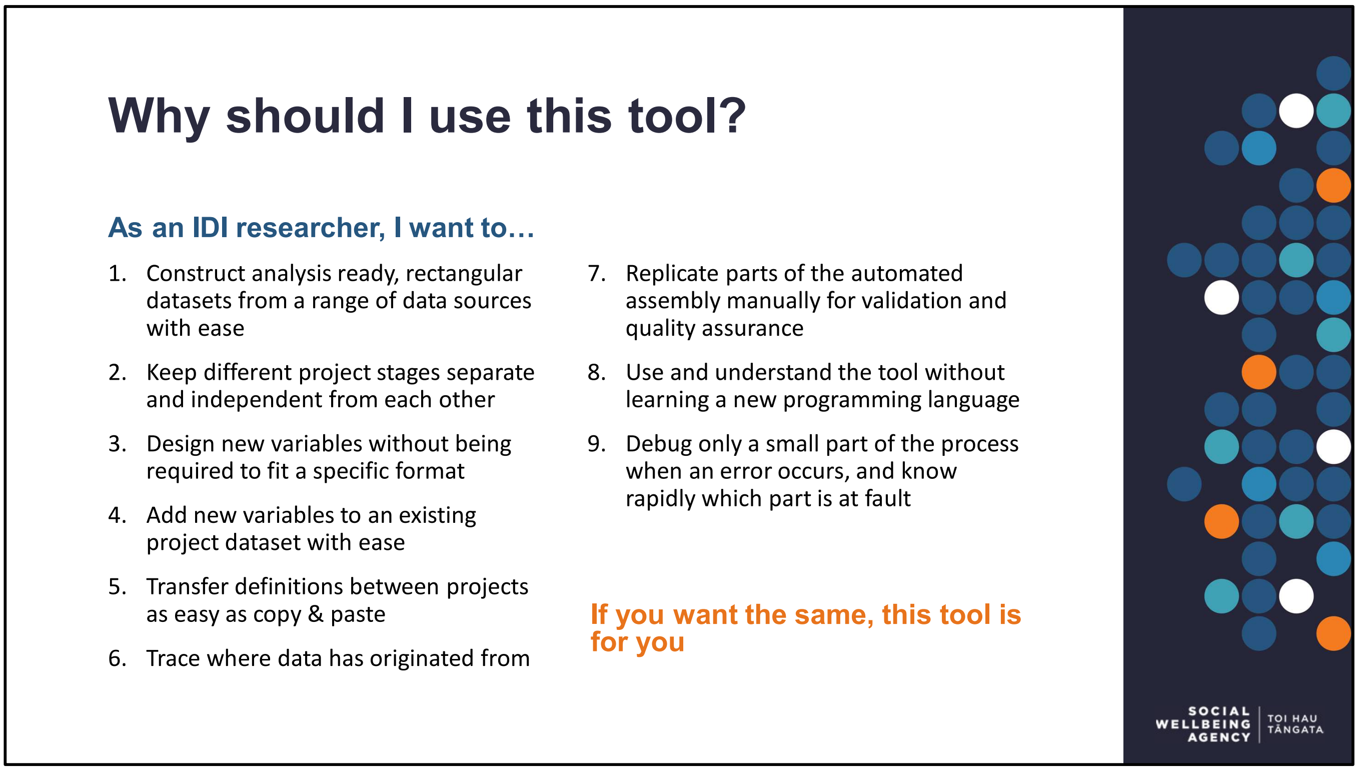
- Move from data wrangling to analytics sooner
- Test ideas faster
- Create your own definitions
- Share and reuse definitions

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

IDI is a rich resource for research. But because the data is gathered for administrative purposes it is not well prepared for research use. Hence significant effort can be required to construct a suitable dataset for research – or researchers limit themselves to only a small part of the IDI.

This is not the first tool developed to make it easier to pull different datasets together. However, where previous tools have seen limited reuse across different projects, the assembly tool has already been applied to multiple projects so we are confident that it offers something of wider value.

The tool was designed within a broader set of practices and patterns. When used in the context of these practices and patterns, it also supports knowledge sharing and provides further acceleration to projects via the reuse of materials between researchers and between projects. If researchers follow similar processes, then components of our research will be more easily shared, reused, and cited by our colleagues. This is discussed in further detail in “Accelerating Dataset Assembly: Primer and guide to the Dataset Assembly Tool”.



Why should I use this tool?

As an IDI researcher, I want to…

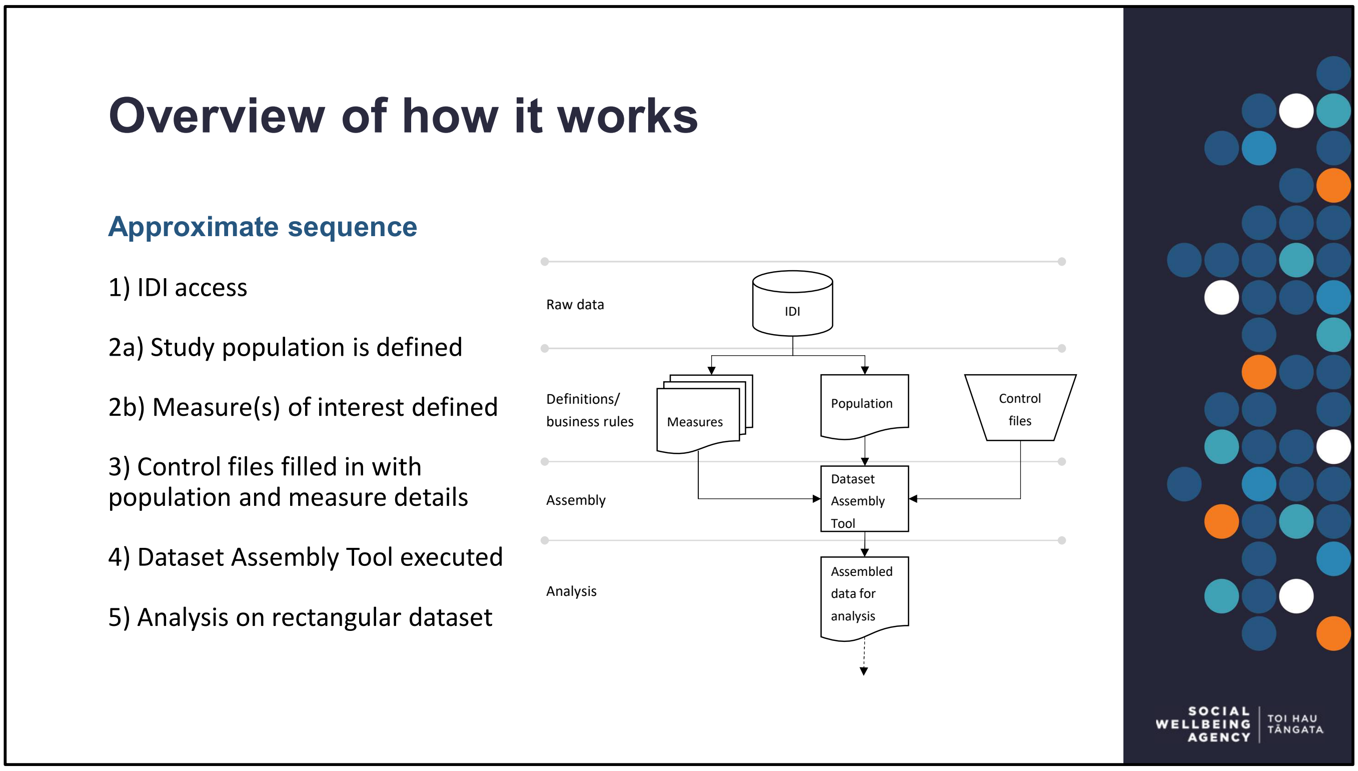
1. Construct analysis ready, rectangular datasets from a range of data sources with ease
2. Keep different project stages separate and independent from each other
3. Design new variables without being required to fit a specific format
4. Add new variables to an existing project dataset with ease
5. Transfer definitions between projects as easy as copy & paste
6. Trace where data has originated from
7. Replicate parts of the automated assembly manually for validation and quality assurance
8. Use and understand the tool without learning a new programming language
9. Debug only a small part of the process when an error occurs, and know rapidly which part is at fault

If you want the same, this tool is for you

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

I assessed several existing tools (including my previous attempts) and identified these nine things I want to do as a researcher based on what needs these tools met, and where there was still frustration. The Dataset Assembly Tool was then built to address each of these needs.

How does the tool meet these needs?
1. Accepts lists of inputs, and outputs rectangular dataset
2. Definitions must run before assembly, most efficient to re-run assembly when adding new definitions than appending them on
3. Tool input adapts to different formats
4. Keeping assembly fast so re-assembling is easy
5. Definitions are separate from assembly
6. Input control files act as record of what was assembled & from where
7. Assembly tool uses a single core query – query is saved to disc for inspection, and it easy to duplicate
8. Tool written in R, but control files are Excel (everyone can use Excel)
9. Multiple places of error checks with sensible messages



Overview of how it works

Approximate sequence

1) IDI access

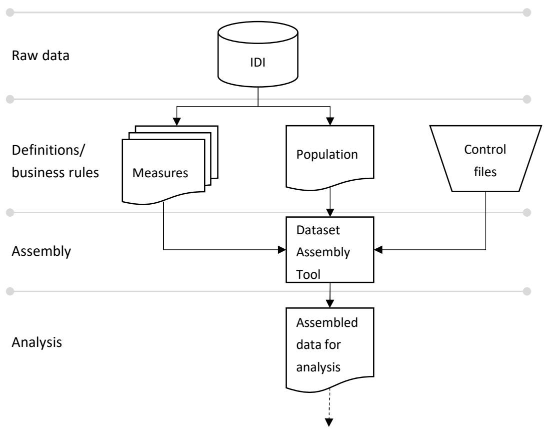
2a) Study population is defined

2b) Measure(s) of interest defined

3) Control files filled in with population and measure details

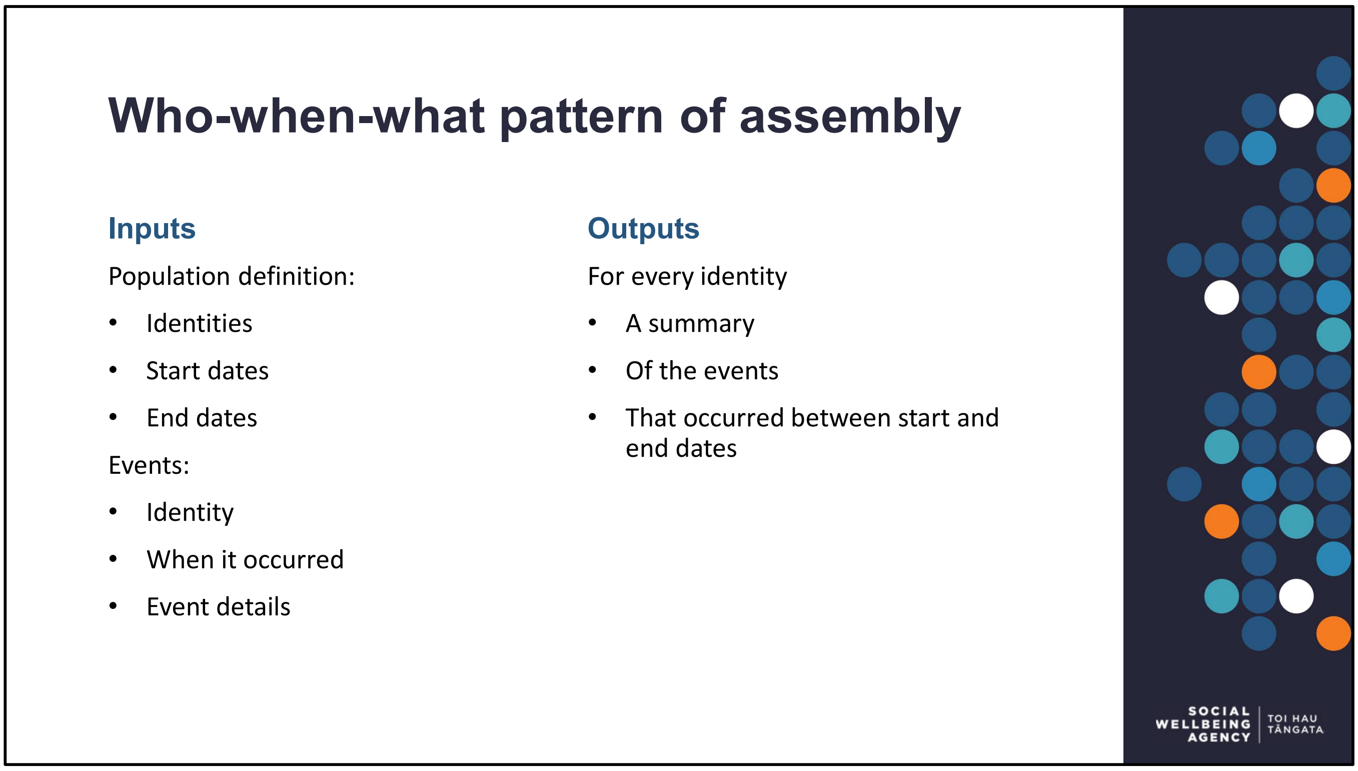
4) Dataset Assembly Tool executed

5) Analysis on rectangular dataset



1. Access to the raw data/potential
2. Researchers define their study population and the events/measures that are of interest to their research.
   - All of these definitions are independent from each other.
   - Definitions could be copied in from other projects
   - No way to avoid having to define what you are interested in
3. Excel control files are populated
   - Includes where to get the definition from
   - How to summarise/prepare it
   - Control files becomes your initial data dictionary
4. Dataset Assembly Tool is executed
   - Control files are read & checked
   - Every measure is created for every population
5. Assembled data is available for research
   - (start with data cleaning – e.g. how should missing values be handled)

More details and a worked example can be found in “Accelerating Dataset Assembly: Primer and guide to the Dataset Assembly Tool”. Also the assembly tool code contains an example using IRD data.



Who-when-what pattern of assembly

Inputs

Population definition:

- Identities
- Start dates
- End dates

Events:

- Identity
- When it occurred
- Event details

Outputs

For every identity

- A summary
- Of the events
- That occurred between start and end dates

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

The assembly tool has been designed to carry out the most common dataset assembly pattern we have observed. This pattern takes two inputs:

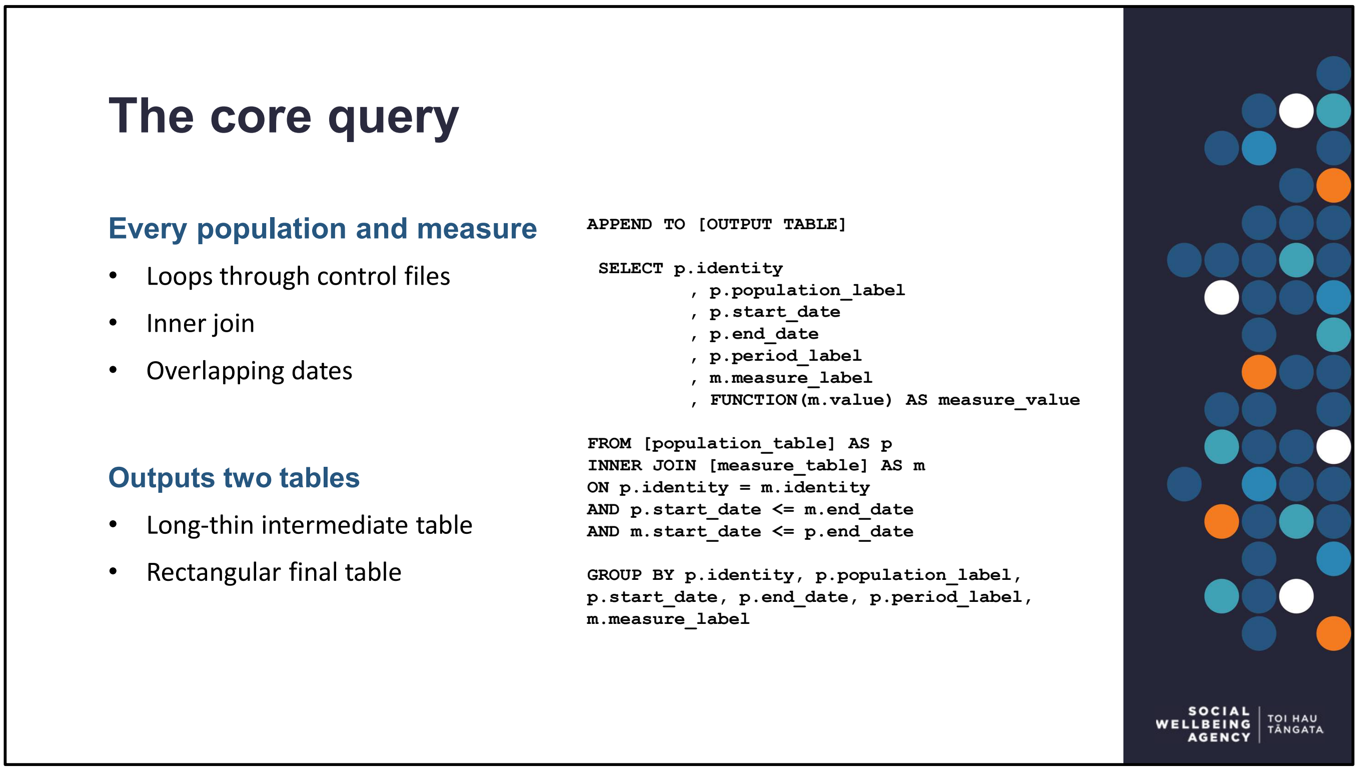
1. Population definition and a date range of interest
2. A (series of) event definitions with identities, dates and details

Defining each of these separately is usually straightforward.

For example:

1. Population = A list of school leavers, from when they graduated until one-year later
2. Event = Income from wages & salaries reported to IRD

These could be combined to output
For each school leaver, total earnings in the year following graduation.



The core query

Every population and measure

- Loops through control files
- Inner join
- Overlapping dates

Outputs two tables

- Long-thin intermediate table
- Rectangular final table

```
APPEND TO [OUTPUT TABLE]

 SELECT p.identity
         , p.population_label
         , p.start_date
         , p.end_date
         , p.period_label
         , m.measure_label
         , FUNCTION(m.value) AS measure_value

FROM [population_table] AS p
INNER JOIN [measure_table] AS m
ON p.identity = m.identity
AND p.start_date <= m.end_date
AND m.start_date <= p.end_date

GROUP BY p.identity, p.population_label,
p.start_date, p.end_date, p.period_label,
m.measure_label
```

One SQL query does all the main processing. This core query is shown on the right side. It is adapted for each row in the control file, but the overall pattern remains the same.

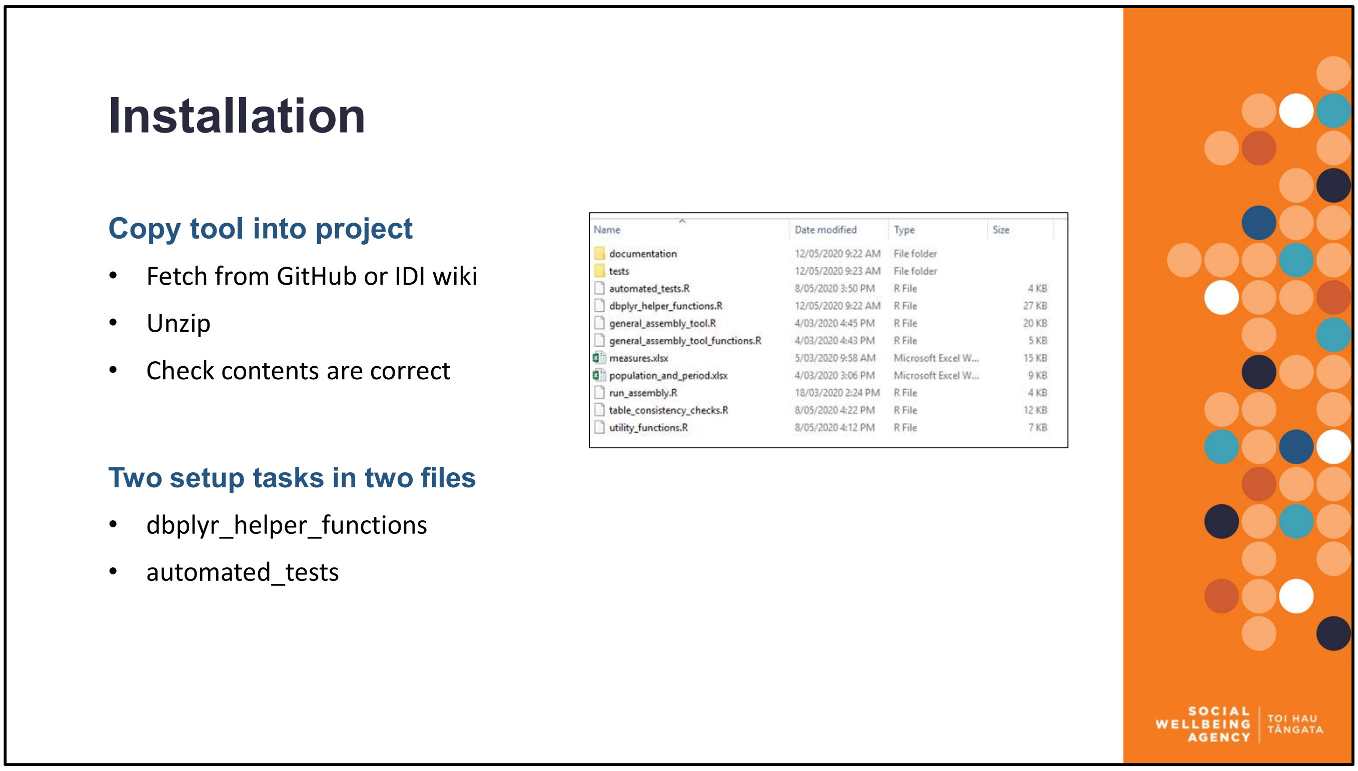
This core query outputs a long, thin table. This intermediate table is designed so it is easy to add all the measures to it using only the core query. This simplifies the tool, making it faster and less vulnerable to errors.
Once all the measures have been appended on to the intermediate table, the assembly tool pivots the table to create a rectangular final table that is ready for research use: One row per observation, and one column per measure.

Because the core query inner joins by identity and overlapping dates, researchers can define their research population completely separate from defining their measures. For my measures, I aim to create the best definition possible (for all people and time periods), knowing that the assembly tool will return only those details that apply to my study population and time period.

First time setup

Before you can use the tool for the first time, some setup is required. This should only be necessary the first time.
Unless you follow these steps the tool will not work.

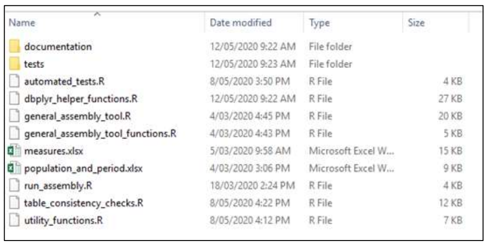
Skip over this section in a short presentation.



Installation

Copy tool into project

- Fetch from GitHub or IDI wiki
- Unzip
- Check contents are correct



Name	Date modified	Type	Size
documentation	12/05/2020 9:22 AM	File folder	
tests	12/05/2020 9:23 AM	File folder	
automated_tests.R	8/05/2020 3:50 PM	R File	4 KB
dbplyr_helper_functions.R	12/05/2020 9:22 AM	R File	27 KB
general_assembly_tool.R	4/03/2020 4:45 PM	R File	20 KB
general_assembly_tool_functions.R	4/03/2020 4:43 PM	R File	5 KB
measures.xlsx	5/03/2020 9:58 AM	Microsoft Excel W...	15 KB
population_and_period.xlsx	4/03/2020 3:06 PM	Microsoft Excel W...	9 KB
run_assembly.R	18/03/2020 2:24 PM	R File	4 KB
table_consistency_checks.R	8/05/2020 4:22 PM	R File	12 KB
utility_functions.R	8/05/2020 4:12 PM	R File	7 KB

Two setup tasks in two files

- dbplyr_helper_functions
- automated_tests

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

The tool is a set of R scripts, plus accompanying documentation, tests and exemplars. You can download these from SWA's GitHub page:
https://github.com/nz-social-wellbeing-agency/dataset_assembly_tool
Or check the IDI wiki for a copy that is already inside the datalab.

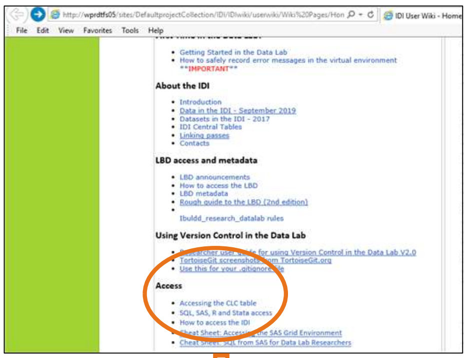
This should arrive bundled in a zip file. Copy into your IDI project and unzip.
The image shows the expected contents of the tool.

Once you have unzipped the tool there are two tasks that need to be completed in two different files.

Set connection details

dbplyr_helper_functions

- SQL connection details are not released from data lab for security
- These can be found on the wiki: Access → “SQL, SAS and R access”
- Enter details into lines 26-28 of dbplyr_helper_functions



http://wprdtfs05/sites/DefaultprojectCollection/IDI/IDIwiki/userwiki/Wiki%20Pages/Hon

IDI User Wiki - Home

- Getting Started in the Data Lab
- How to safely record error messages in the virtual environment **IMPORTANT**

About the IDI

- Introduction
- Data in the IDI - September 2019
- Datasets in the IDI - 2017
- IDI Central Tables
- Linking passes
- Contacts

LBD access and metadata

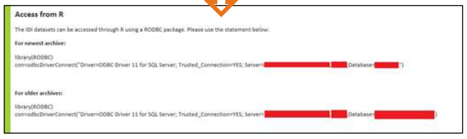
- LBD announcements
- How to access the LBD
- LBD metadata
- Rough guide to the LBD (2nd edition)
- Ibuldd_research_datalab rules

Using Version Control in the Data Lab

- Researcher user guide for using Version Control in the Data Lab V2.0
- TortoiseGit screenshots from TortoiseGit.org
- Use this for your .gitignore file

Access

- Accessing the CLC table
- SQL, SAS, R and Stata access
- How to access the IDI
- Cheat Sheet: Accessing the SAS Grid Environment
- Cheat Sheet: SQL from SAS for Data Lab Researchers



Access from R

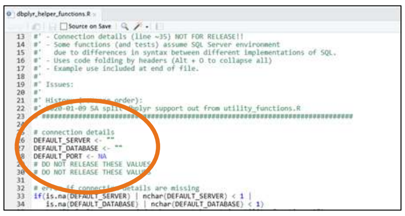
The IDI datasets can be accessed through R using a RODBC package. Please use the statement below:

For newest archive:

```
library(RODBC)
con=odbcDriverConnect("Driver=ODBC Driver 11 for SQL Server; Trusted_Connection=YES; Server=[redacted] [redacted] Database=[redacted]")
```

For older archives:

```
library(RODBC)
con=odbcDriverConnect("Driver=ODBC Driver 11 for SQL Server; Trusted_Connection=YES; Server=[redacted] [redacted] Database=[redacted])
```



dbplyr_helper_functions.R

```
13  #' - Connection details (line ~35) NOT FOR RELEASE!!
14  #' - Some functions (and tests) assume SQL Server environment
15  #'   due to differences in syntax between different implementations of SQL.
16  #' - Uses code folding by headers (Alt + O to collapse all)
17  #' - Example use included at end of file.
18  #'
19  #' Issues:
20  #'
21  #' History (reverse order):
22  #' 2020-01-09 SA split dbplyr support out from utility_functions.R
23  # ################################################################################
24
25  # connection details
26  DEFAULT_SERVER <- ""
27  DEFAULT_DATABASE <- ""
28  DEFAULT_PORT <- NA
29  # DO NOT RELEASE THESE VALUES
30  # DO NOT RELEASE THESE VALUES
31
32  # error if connection details are missing
33  if(is.na(DEFAULT_SERVER) | nchar(DEFAULT_SERVER) < 1 |
34     is.na(DEFAULT_DATABASE) | nchar(DEFAULT_DATABASE) < 1)
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

(Top left image) On the IDI wiki, under the heading “Access”, click the link “SQ:, SAS, R and Stata access”.
(Bottom right) Read off the server name, port number, and database under the “Access from R” heading
(Bottom left) Enter these values into the “connection details” section at the top of dbplyr_helper_functions.R

The final results looks like:

DEFAULT_SERVER <- “[server.name.bond]”
DEFAULT_DATEBASE <- “[database.name.james]”
DEFAULT_PORT <- 007

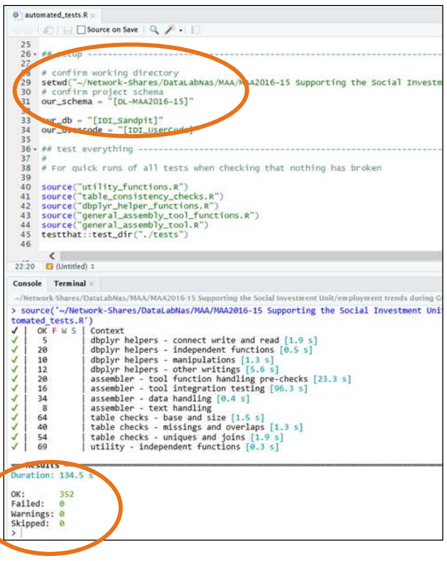
Run automated tests

automated_tests

Automated tests confirm that the tool is setup correctly and ready to go

- Set the directory (line 29)
- Set your project schema (line 31)
- Run / source the script

- Tests take at most 5 minutes
- Output reports tests passed



```
25
26 ## Setup -------------------------------------------------------------
27
28 # confirm working directory
29 setwd("~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investm
30 # confirm project schema
31 our_schema = "[DL-MAA2016-15]"
32
33 our_db = "[IDI_Sandpit]"
34 our_usercode = "[IDI_UserCode]"
35
36 ## test everything ------------------------------------------------------
37 #
38 # For quick runs of all tests when checking that nothing has broken
39
40 source("utility_functions.R")
41 source("table_consistency_checks.R")
42 source("dbplyr_helper_functions.R")
43 source("general_assembly_tool_functions.R")
44 source("general_assembly_tool.R")
45 testthat::test_dir("./tests")
46
```

```
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during G
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Uni
tomated_tests.R')
✓ |  OK F W S | Context
✓ |   5       | dbplyr helpers - connect write and read [1.9 s]
✓ |  20       | dbplyr helpers - independent functions [0.5 s]
✓ |  10       | dbplyr helpers - manipulations [1.3 s]
✓ |  12       | dbplyr helpers - other writings [5.6 s]
✓ |  20       | assembler - tool function handling pre-checks [23.3 s]
✓ |  16       | assembler - tool integration testing [96.3 s]
✓ |  34       | assembler - data handling [0.4 s]
✓ |   8       | assembler - text handling
✓ |  64       | table checks - base and size [1.5 s]
✓ |  40       | table checks - missings and overlaps [1.3 s]
✓ |  54       | table checks - uniques and joins [1.9 s]
✓ |  69       | utility - independent functions [0.3 s]

══ Results ═══════════════════════════════════════════════════════════════
Duration: 134.5 s

OK:       352
Failed:   0
Warnings: 0
Skipped:  0
>
```

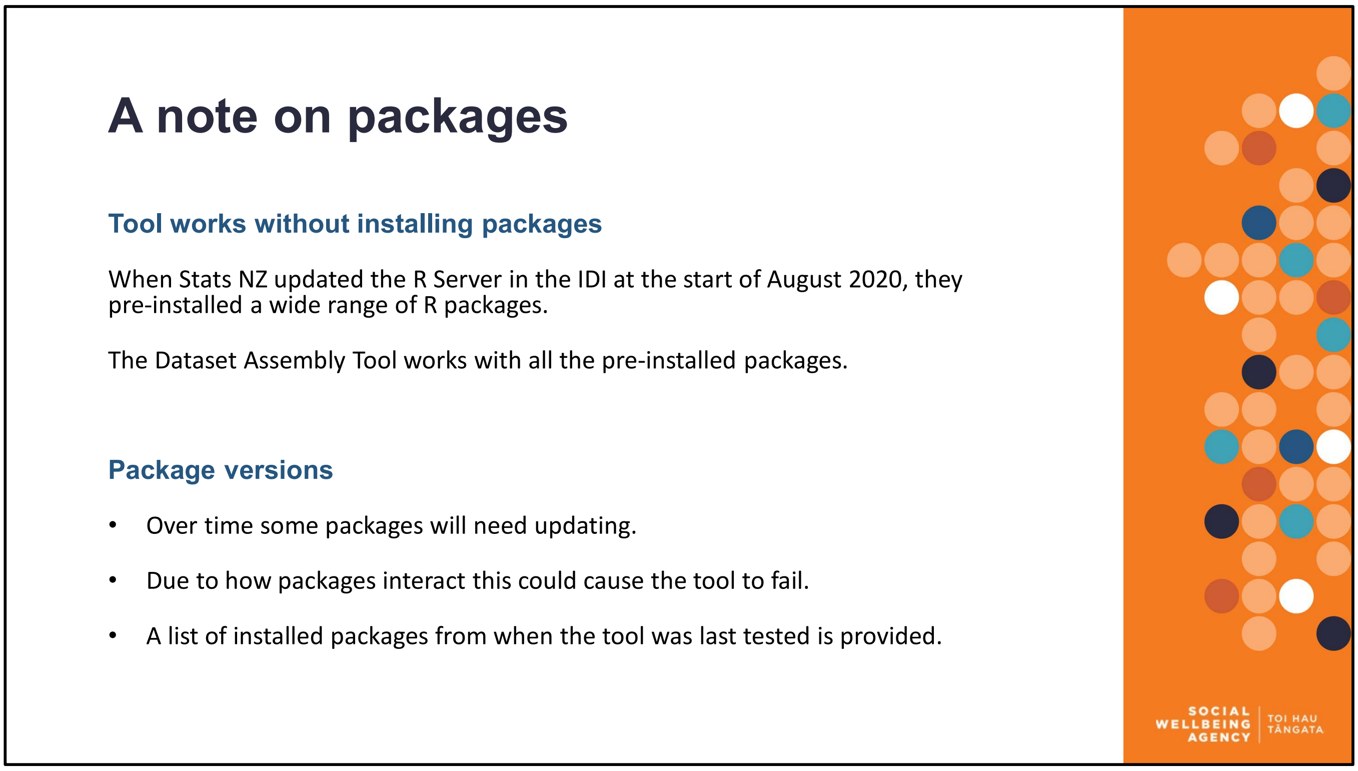
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

The assembly tool is covered by unit record tests. These are tests that check all of the functions of the tool (and associated code) are performing correctly. They include copying a worked example onto the SQL server and checking that it returns the correct output.

All the tests and resources required to run them are found in the “tests” folder. However, all you need to access is “automated_tests.R”.
This script will run all the automated tests and output a summary. If all the tests pass then you have confirmed that the tool is setup and running correctly.

Top half of image is the testing script. The two lines that need to be updated are circled.
Bottom half of image is the testing output:
- All the tests run
- Whether tests passed (OK), failed (F), gave warnings (W) or were skipped (S)
- Circled is the final summary showing all tests were passed.

The tests are best run in a new/empty session. If errors occur / the tests fail, then see the section at the end of this presentation of what to do when things go wrong.



A note on packages

Tool works without installing packages

When Stats NZ updated the R Server in the IDI at the start of August 2020, they pre-installed a wide range of R packages.

The Dataset Assembly Tool works with all the pre-installed packages.

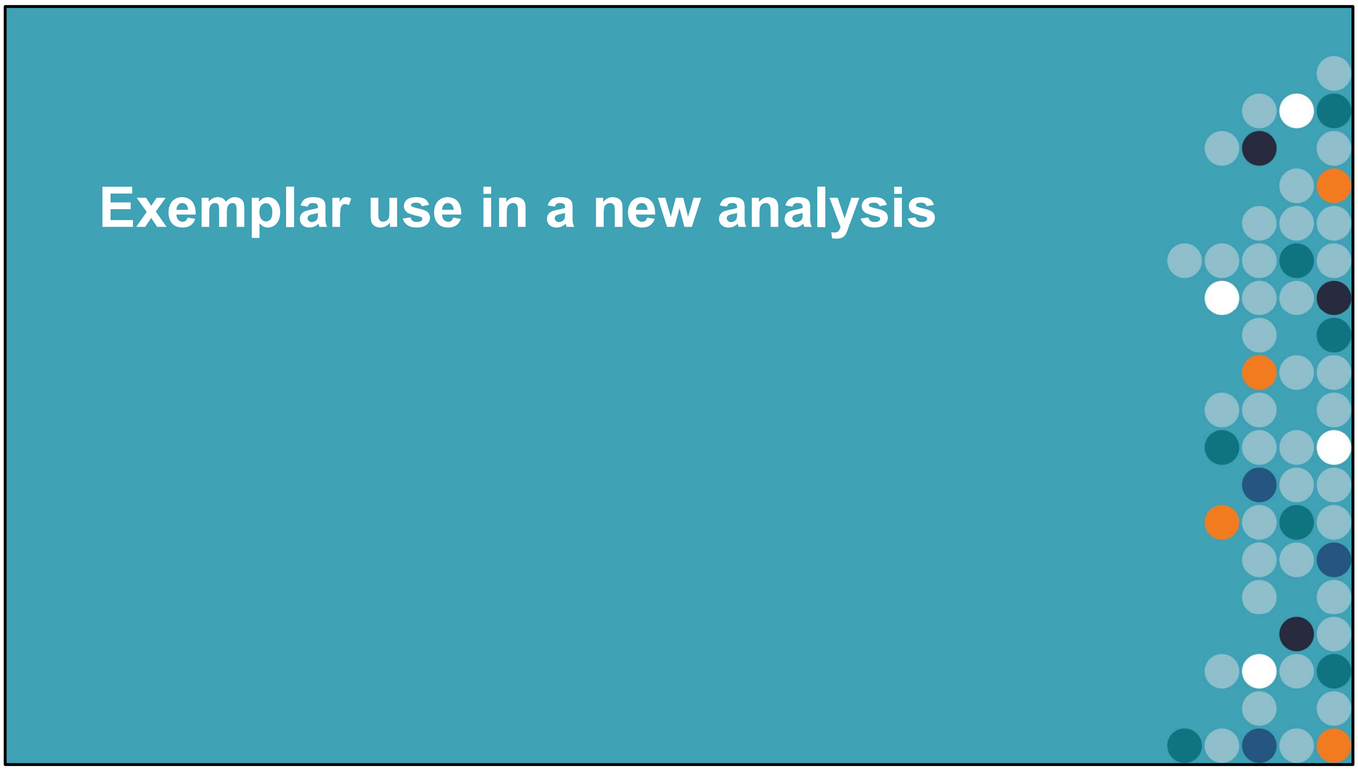
Package versions

- Over time some packages will need updating.
- Due to how packages interact this could cause the tool to fail.
- A list of installed packages from when the tool was last tested is provided.

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

We also provide a list of the packages & versions installed with our working configuration of the tool. Some users with other versions of packages installed may experience errors. Ensuring that your package versions are consistent with ours reduces the risk of this. As Stats NZ has already installed a range of default packages, it should generally be sufficient to uninstall any version you have installed yourself and rely on the version Stats NZ provide to all users.

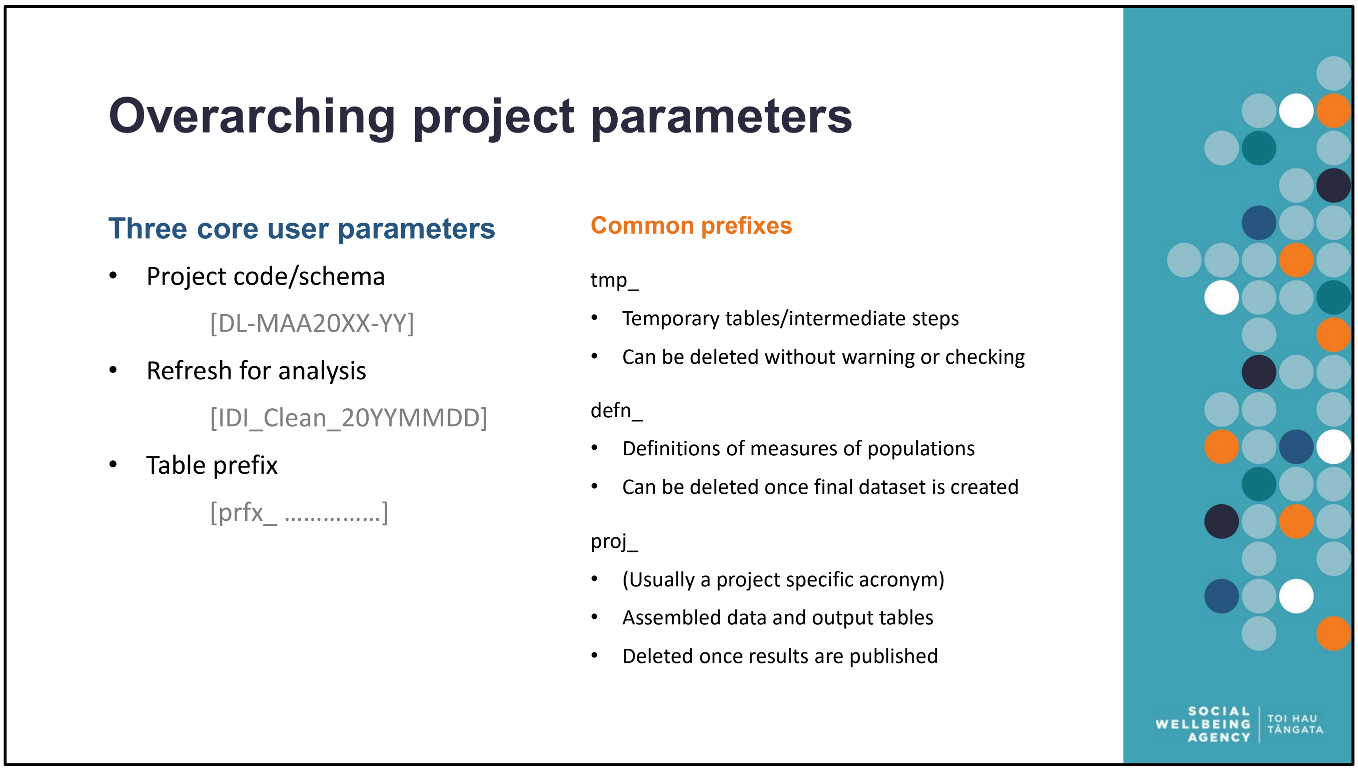
Package & version differences are more likely to cause warnings than errors. E.g. prior to August 2020 the tool would produce warnings from the ‘rlang’ package about some commands being depreciated. Warnings are unlikely to prevent the tool from running.



Exemplar use in a new analysis

This section covers an example of using the tool. It demonstrates all the key workings that researchers will need.

As part of designing this example, I went from an empty project to a complete dataset in half a day.



Overarching project parameters

Three core user parameters

- Project code/schema

  [DL-MAA20XX-YY]
- Refresh for analysis

  [IDI_Clean_20YYMMDD]
- Table prefix

  [prfx_ ……………]

Common prefixes

tmp_

- Temporary tables/intermediate steps
- Can be deleted without warning or checking

defn_

- Definitions of measures of populations
- Can be deleted once final dataset is created

proj_

- (Usually a project specific acronym)
- Assembled data and output tables
- Deleted once results are published

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Several parameters turn up in many places throughout a research project.
Deciding these once at the start of the project helps ensure consistency throughout the project.

It is the user's responsibility to ensure that these parameters are consistent throughout their analysis.
(The tool does not check that you have used the same refresh or schema name throughout. The user is responsible for ensuring consistent parameters are used.)

A table prefix is optional, but highly recommended.
Because tables are sorted alphabetically in SQL Server, using a prefix keeps related tables together. It also makes purpose of table clear. Aids identification and removal of unneeded tables – being a good steward to data lab resources.

As an example of a project specific prefix, I used "chh_" as the prefix for my analysis into "constructing households"

Define study population

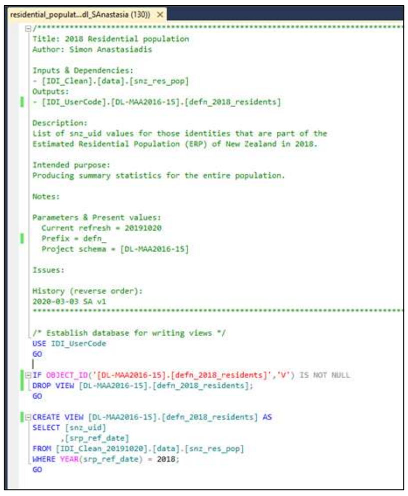
Components

Compulsory:

- Who

Optional:

- When
- Label for identity
- Label for time period

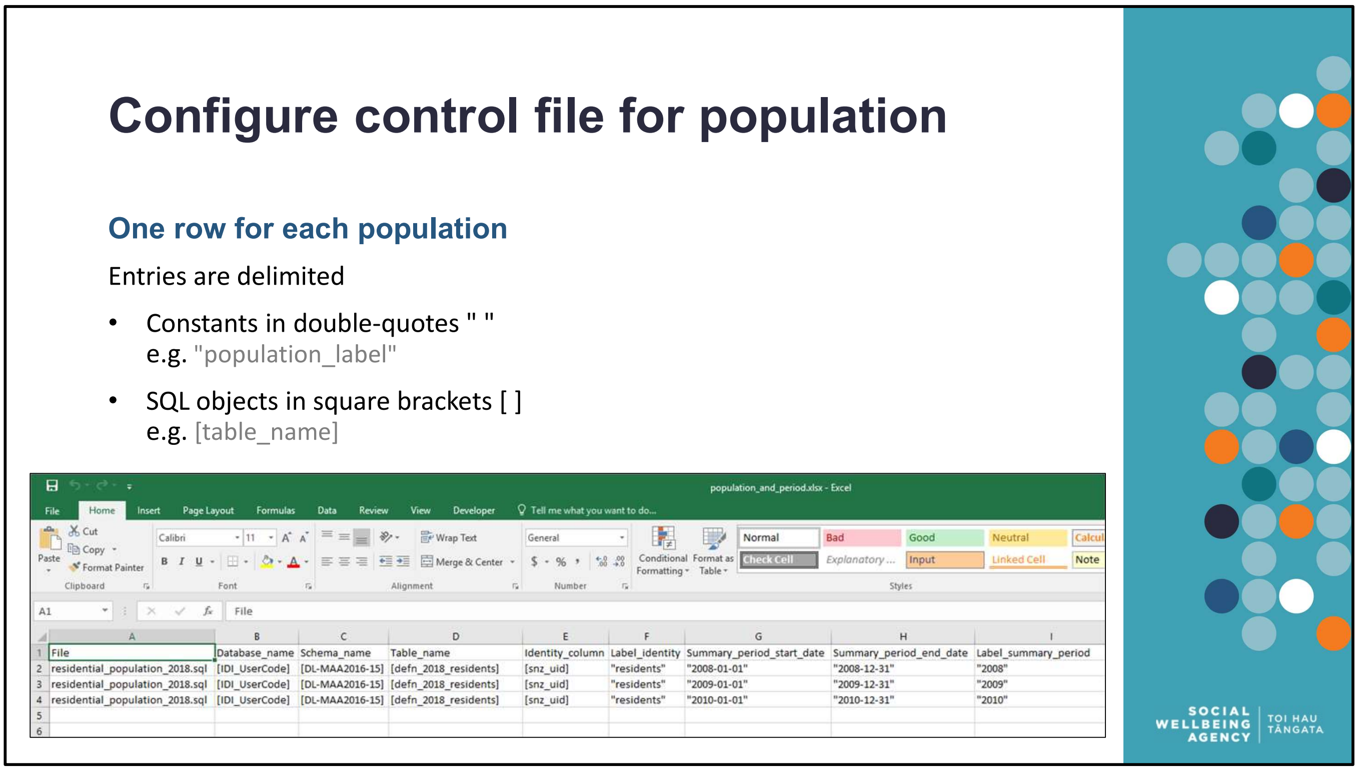

residential_populat...dl_SAnastasia (130))

```sql
/*******************************************************************************
Title: 2018 Residential population
Author: Simon Anastasiadis

Inputs & Dependencies:
- [IDI_Clean].[data].[snz_res_pop]
Outputs:
- [IDI_UserCode].[DL-MAA2016-15].[defn_2018_residents]

Description:
List of snz_uid values for those identities that are part of the
Estimated Residential Population (ERP) of New Zealand in 2018.

Intended purpose:
Producing summary statistics for the entire population.

Notes:

Parameters & Present values:
  Current refresh = 20191020
  Prefix = defn_
  Project schema = [DL-MAA2016-15]

Issues:

History (reverse order):
2020-03-03 SA v1
*******************************************************************************

/* Establish database for writing views */
USE IDI_UserCode
GO

IF OBJECT_ID('[DL-MAA2016-15].[defn_2018_residents]','V') IS NOT NULL
DROP VIEW [DL-MAA2016-15].[defn_2018_residents];
GO

CREATE VIEW [DL-MAA2016-15].[defn_2018_residents] AS
SELECT [snz_uid]
      ,[srp_ref_date]
FROM [IDI_Clean_20191020].[data].[snz_res_pop]
WHERE YEAR(srp_ref_date) = 2018;
GO
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

An SQL script must be written that defines your population of interest. It must contain 'who' is in the study population.
If when each person is being studies differs then the table should also contain when.
If the same person is studied more than once (e.g. before and after groups), or in more than one time period, you may also wish to provide labels for the identity (or group of identities) or the time period.

More than one study population can be defined (E.g. one definition for treatment, and one definition for control), each in its own file.



Configure control file for population

One row for each population

Entries are delimited

- Constants in double-quotes " "
  e.g. "population_label"
- SQL objects in square brackets [ ]
  e.g. [table_name]

population_and_period.xlsx - Excel

	A	B	C	D	E	F	G	H	I
1	File	Database_name	Schema_name	Table_name	Identity_column	Label_identity	Summary_period_start_date	Summary_period_end_date	Label_summary_period
2	residential_population_2018.sql	[IDI_UserCode]	[DL-MAA2016-15]	[defn_2018_residents]	[snz_uid]	"residents"	"2008-01-01"	"2008-12-31"	"2008"
3	residential_population_2018.sql	[IDI_UserCode]	[DL-MAA2016-15]	[defn_2018_residents]	[snz_uid]	"residents"	"2009-01-01"	"2009-12-31"	"2009"
4	residential_population_2018.sql	[IDI_UserCode]	[DL-MAA2016-15]	[defn_2018_residents]	[snz_uid]	"residents"	"2010-01-01"	"2010-12-31"	"2010"
5									
6									

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Enter the details of each population in the table.
All columns other than File are compulsory. Every cell must be populated. Compulsory cells must be delimited.

Columns:
- Database, schema and table names give the SQL location where the table is saved
- Identity is the column that contains your individual identifiers (will most often be [snz_uid] but could take other values, like meshblock codes if needed)
- Summary period start and end date define the period of time of interest for our population (Format: “YYYY-MM-DD”, e.g. “2020-12-31”)
- Identity labels are assigned to the identities
- Summary period labels are assigned to the periods

The example control file, creates a panel of data for the same population for three years.

Details on each column of the population file

1. File (optional) – Recommended you record the file name of the SQL script that creates your study population.

2. Database Name (required, sql object only) – The name of the SQL database

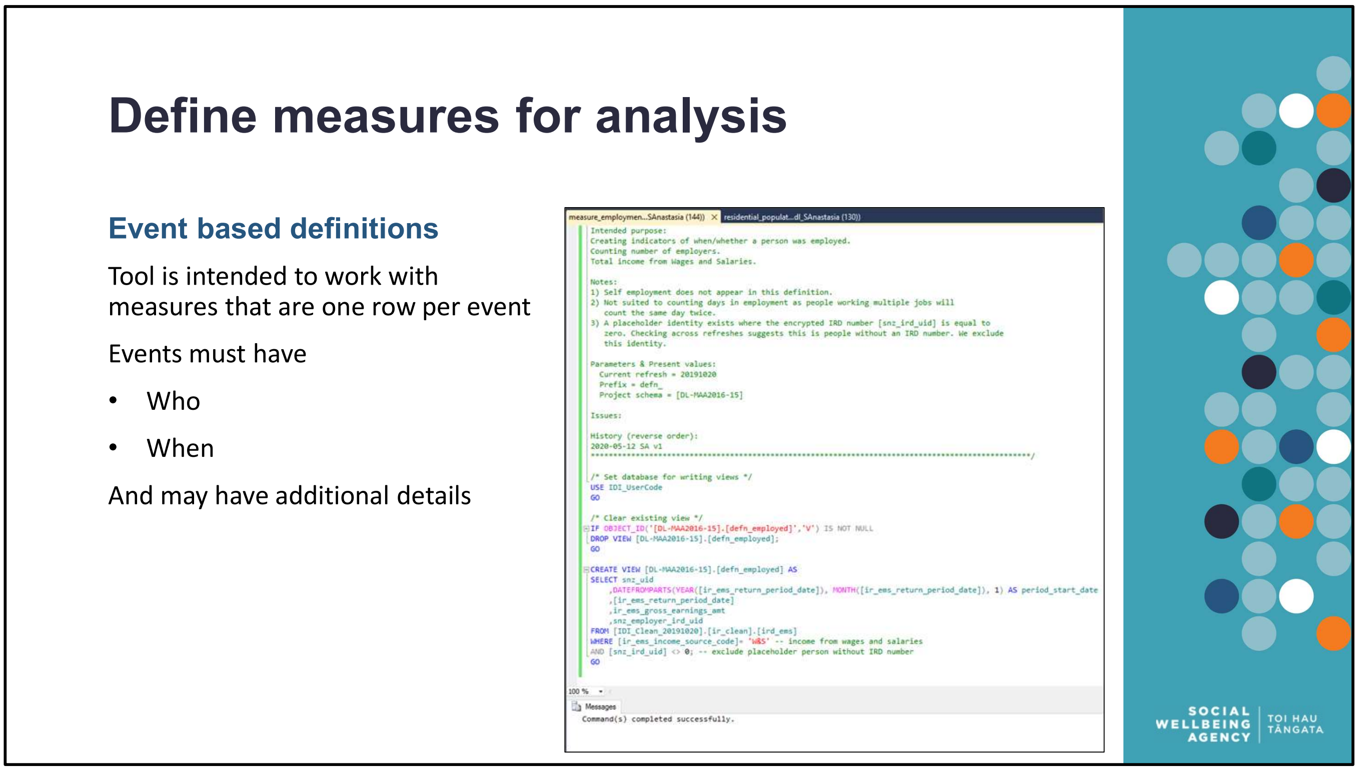
where the study population is stored. Most likely [IDI_Clean_XXXX] or [IDI_Sandpit] or [IDI_UserCode].

3. Schema Name (required, sql object only) – The name of the schema where the study population is stored. Most likely a data-type scheme, e.g. [moe_clean], or a project schema, e.g. [DL-MAA20XX-XX].
4. Table Name (required, sql object only) – The name of the table/view where the subject population is stored.
5. Identity Column (required, sql object or constant) – The name of the column that contains the unique identifier for the population. This can be an individual identifier, e.g. [snz_uid], or some other identifier, e.g. [snz_xxx_uid], [mb_2018], [school_id].
6. Label Identity (required, sql object or constant) – Used to label your study population, e.g. “treat” and “control”. Use a placeholder like “person” if unneeded.
7. Summary Period Start Date (required, sql object or constant) – Start date for time of interest for the subject population. If entering constants, the data must be given as “YYYY-MM-DD” e.g. “2020-01-01” as the start of 2020.
8. Summary Period End Date (required, sql object or constant) – End date for time of interest for the subject population. If entering constants, the data must be given as “YYYY-MM-DD” e.g. “2020-12-31” as the end of 2020.
9. Label Summary Period (required, sql object or constant) – Used to label your study periods, e.g. “before” and “after”. Use a placeholder like the study year “2018“ if unneeded.

The assembly tool does not require the input or output to be individual people. Identity_column could contain geographic or business identifiers. More details on how different types of identities can be used are discussed below under “Non-individual identities”.

The start and end date are used to filter all the measures. Only measures that overlap with the summary period for the population will be included in the output. So start date “2020-01-01” and end date “2020-12-31” will exclude measures that sit outside the year 2020 (e.g. income, hospital visits, benefit receipt that ended before 2020, or that started after 2020 will be excluded).

The correct format for dates is YYYY-MM-DD. Other date formats may not result in an error but could lead to incorrect output (for example, depending on the computer’s settings DD-MM-YYYY and MM-DD-YYYY dates can be treated as the wrong type).



Define measures for analysis

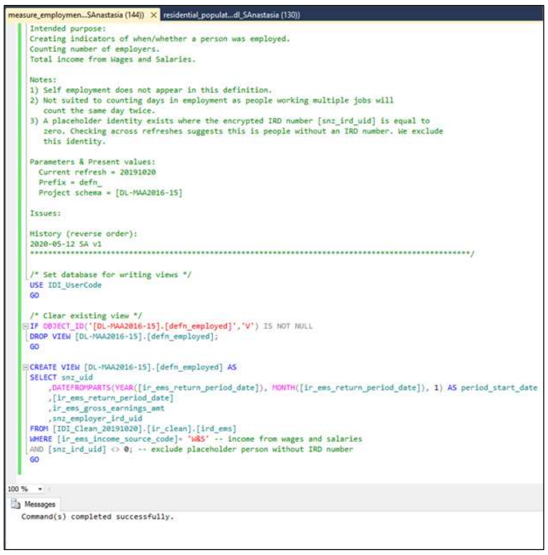
Event based definitions

Tool is intended to work with measures that are one row per event

Events must have

- Who
- When

And may have additional details



measure_employmen...SAnastasia (144))
residential_populat...dl_SAnastasia (130))

```
Intended purpose:
Creating indicators of when/whether a person was employed.
Counting number of employers.
Total income from Wages and Salaries.

Notes:
1) Self employment does not appear in this definition.
2) Not suited to counting days in employment as people working multiple jobs will
   count the same day twice.
3) A placeholder identity exists where the encrypted IRD number [snz_ird_uid] is equal to
   zero. Checking across refreshes suggests this is people without an IRD number. We exclude
   this identity.

Parameters & Present values:
  Current refresh = 20191020
  Prefix = defn_
  Project schema = [DL-MAA2016-15]

Issues:

History (reverse order):
2020-05-12 SA v1
**************************************************************************************************/

/* Set database for writing views */
USE IDI_UserCode
GO

/* Clear existing view */
IF OBJECT_ID('[DL-MAA2016-15].[defn_employed]','V') IS NOT NULL
DROP VIEW [DL-MAA2016-15].[defn_employed];
GO

CREATE VIEW [DL-MAA2016-15].[defn_employed] AS
SELECT snz_uid
    ,DATEFROMPARTS(YEAR([ir_ems_return_period_date]), MONTH([ir_ems_return_period_date]), 1) AS period_start_date
    ,[ir_ems_return_period_date]
    ,ir_ems_gross_earnings_amt
    ,snz_employer_ird_uid
FROM [IDI_Clean_20191020].[ir_clean].[ird_ems]
WHERE [ir_ems_income_source_code]= 'W&S' -- income from wages and salaries
AND [snz_ird_uid] <> 0; -- exclude placeholder person without IRD number
GO
```

```
Command(s) completed successfully.
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

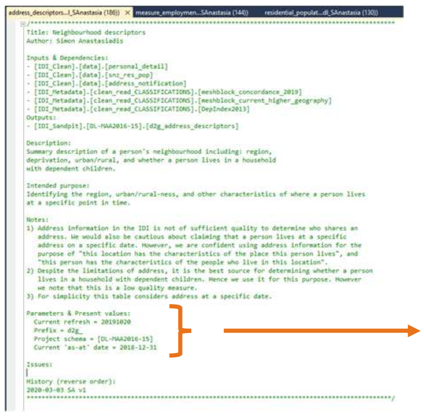
Example on RHS is of an employment event. The details of this event include who, when, income paid, employer's identity.

The tool works best when the measures are from tables that are one row per event. It helps that most IDI tables are already one row per event (e.g. hospital discharge, monthly tax summary, school enrolment). But raw table events may not be the events we want to study, and so we make new events from old ones (e.g. hospital discharge event → knee replacement surgery event; OR school enrolment → studying at university event).

I recommend, defining simple measures as Views in IDI_UserCode.
And defining more complex measures as Tables in IDI_Sandpit.
For some very simple events you may be able to use raw data directly.

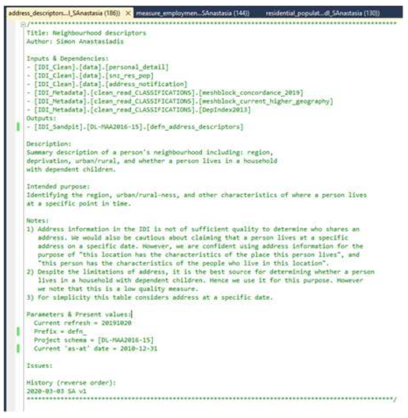
Reuse measures for analysis

Old definition


address_descriptors...l_SAnastasia (186))
measure_employmen...SAnastasia (144))
residential_populat...dl_SAnastasia (130))

```
/*********************************************************************************************
Title: Neighbourhood descriptors
Author: Simon Anastasiadis

Inputs & Dependencies:
- [IDI_Clean].[data].[personal_detail]
- [IDI_Clean].[data].[snz_res_pop]
- [IDI_Clean].[data].[address_notification]
- [IDI_Metadata].[clean_read_CLASSIFICATIONS].[meshblock_concordance_2019]
- [IDI_Metadata].[clean_read_CLASSIFICATIONS].[meshblock_current_higher_geography]
- [IDI_Metadata].[clean_read_CLASSIFICATIONS].[DepIndex2013]
Outputs:
- [IDI_Sandpit].[DL-MAA2016-15].[d2g_address_descriptors]

Description:
Summary description of a person's neighbourhood including: region,
deprivation, urban/rural, and whether a person lives in a household
with dependent children.

Intended purpose:
Identifying the region, urban/rural-ness, and other characteristics of where a person lives
at a specific point in time.

Notes:
1) Address information in the IDI is not of sufficient quality to determine who shares an
   address. We would also be cautious about claiming that a person lives at a specific
   address on a specific date. However, we are confident using address information for the
   purpose of "this location has the characteristics of the place this person lives", and
   "this person has the characteristics of the people who live in this location".
2) Despite the limitations of address, it is the best source for determining whether a person
   lives in a household with dependent children. Hence we use it for this purpose. However
   we note that this is a low quality measure.
3) For simplicity this table considers address at a specific date.

Parameters & Present values:
  Current refresh = 20191020
  Prefix = d2g_
  Project schema = [DL-MAA2016-15]
  Current 'as-at' date = 2018-12-31

Issues:

History (reverse order):
2020-03-03 SA v1
*********************************************************************************************/
```

Revised definition


address_descriptors...l_SAnastasia (186))
measure_employmen...SAnastasia (144))
residential_populat...dl_SAnastasia (130))

```
/*********************************************************************************************
Title: Neighbourhood descriptors
Author: Simon Anastasiadis

Inputs & Dependencies:
- [IDI_Clean].[data].[personal_detail]
- [IDI_Clean].[data].[snz_res_pop]
- [IDI_Clean].[data].[address_notification]
- [IDI_Metadata].[clean_read_CLASSIFICATIONS].[meshblock_concordance_2019]
- [IDI_Metadata].[clean_read_CLASSIFICATIONS].[meshblock_current_higher_geography]
- [IDI_Metadata].[clean_read_CLASSIFICATIONS].[DepIndex2013]
Outputs:
- [IDI_Sandpit].[DL-MAA2016-15].[defn_address_descriptors]

Description:
Summary description of a person's neighbourhood including: region,
deprivation, urban/rural, and whether a person lives in a household
with dependent children.

Intended purpose:
Identifying the region, urban/rural-ness, and other characteristics of where a person lives
at a specific point in time.

Notes:
1) Address information in the IDI is not of sufficient quality to determine who shares an
   address. We would also be cautious about claiming that a person lives at a specific
   address on a specific date. However, we are confident using address information for the
   purpose of "this location has the characteristics of the place this person lives", and
   "this person has the characteristics of the people who live in this location".
2) Despite the limitations of address, it is the best source for determining whether a person
   lives in a household with dependent children. Hence we use it for this purpose. However
   we note that this is a low quality measure.
3) For simplicity this table considers address at a specific date.

Parameters & Present values:
  Current refresh = 20191020
  Prefix = defn_
  Project schema = [DL-MAA2016-15]
  Current 'as-at' date = 2018-12-31

Issues:

History (reverse order):
2020-03-03 SA v1
*********************************************************************************************/
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

This is also a good opportunity to reuse/revise previous definitions.
In the example above, we update some of the parameters of an old definition so it can be reused as a new definition.

Both images show the meta-data header I use for my definitions.
One component of this header is a list of parameters that are particular to an analysis (e.g. project prefix, IDI refresh, target date). This makes find & replace an easy means of reusing entire definitions.

When doing this take care to check your inputs for consistency. The user is responsible for ensuring the correct parameters (including choice of IDI refresh) are used everywhere. You will get odd results if you reuse definitions that refer to a different IDI refresh.

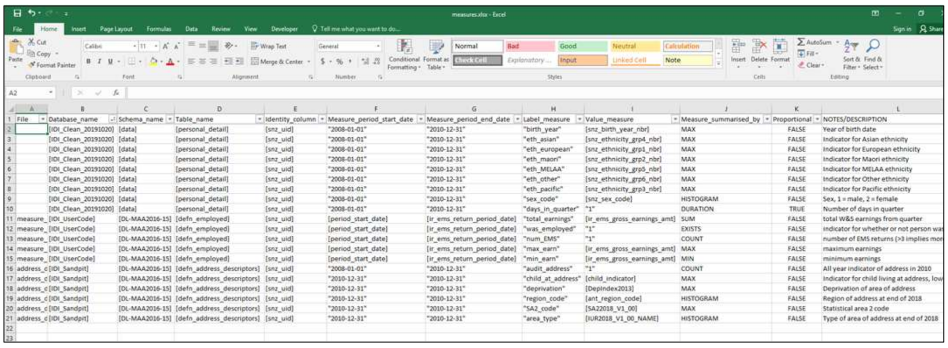
Configure control file for measures

One row for each output

A single event can produce multiple outputs

Same delimiting as population control

File becomes your starting data dictionary



File	Database_name	Schema_name	Table_name	Identity_column	Measure_period_start_date	Measure_period_end_date	Label_measure	Value_measure	Measure_summarised_by	Proportional	NOTES/DESCRIPTION
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_year"	[snz_birth_year_nbr]	MAX	FALSE	Year of birth date
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_asian"	[snz_ethnicity_grp4_nbr]	MAX	FALSE	Indicator for Asian ethnicity
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_european"	[snz_ethnicity_grp1_nbr]	MAX	FALSE	Indicator for European ethnicity
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_maori"	[snz_ethnicity_grp2_nbr]	MAX	FALSE	Indicator for Maori ethnicity
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_MELAA"	[snz_ethnicity_grp5_nbr]	MAX	FALSE	Indicator for MELAA ethnicity
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_other"	[snz_ethnicity_grp6_nbr]	MAX	FALSE	Indicator for Other ethnicity
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_pacific"	[snz_ethnicity_grp3_nbr]	MAX	FALSE	Indicator for Pacific ethnicity
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"sex_code"	[snz_sex_code]	HISTOGRAM	FALSE	Sex, 1 = male, 2 = female
	[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"days_in_quarter"	"1"	DURATION	TRUE	Number of days in quarter
measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"total_earnings"	[ir_ems_gross_earnings_amt]	SUM	FALSE	total W&S earnings from quarter
measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"was_employed"	"1"	EXISTS	FALSE	indicator for whether or not person wa
measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"num_EMS"	"1"	COUNT	FALSE	number of EMS returns (>3 implies mor
measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"max_earn"	[ir_ems_gross_earnings_amt]	MAX	FALSE	maximum earnings
measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"min_earn"	[ir_ems_gross_earnings_amt]	MIN	FALSE	minimum earnings
address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2008-01-01"	"2010-12-31"	"audit_address"	"1"	COUNT	FALSE	All year indicator of address in 2010
address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"child_at_address"	[child_indicator]	MAX	FALSE	Indicator for child living at address, low
address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"deprivation"	[DepIndex2013]	MAX	FALSE	Deprivation of area of address
address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"region_code"	[ant_region_code]	HISTOGRAM	FALSE	Region of address at end of 2018
address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"SA2_code"	[SA22018_V1_00]	MAX	FALSE	Statistical area 2 code
address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"area_type"	[IUR2018_V1_00_NAME]	HISTOGRAM	FALSE	Type of area of address at end of 2018

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Enter the details of each measure you want in your output.
All columns other than File and Notes/description are compulsory.

Configuration of measure file is very similar to the population file. Please refer to notes from configuration of population file.

Columns:
- Database, schema and table names give the SQL location where the table is saved
- Identity is the column that contains individual identifiers
- Measure period start and end date define the period of time of the event (Format: "YYYY-MM-DD", e.g. "2020-12-31")
- Label measure is the labels your output will have (recommended to use underscores instead of spaces)
- The value your output will have is determined by "Measure_summarised_by" applied to "Value_measure"
- Proportional is used to pro rate to the intersection between the summary period and the measure period. It is especially important when working with the duration summary type.

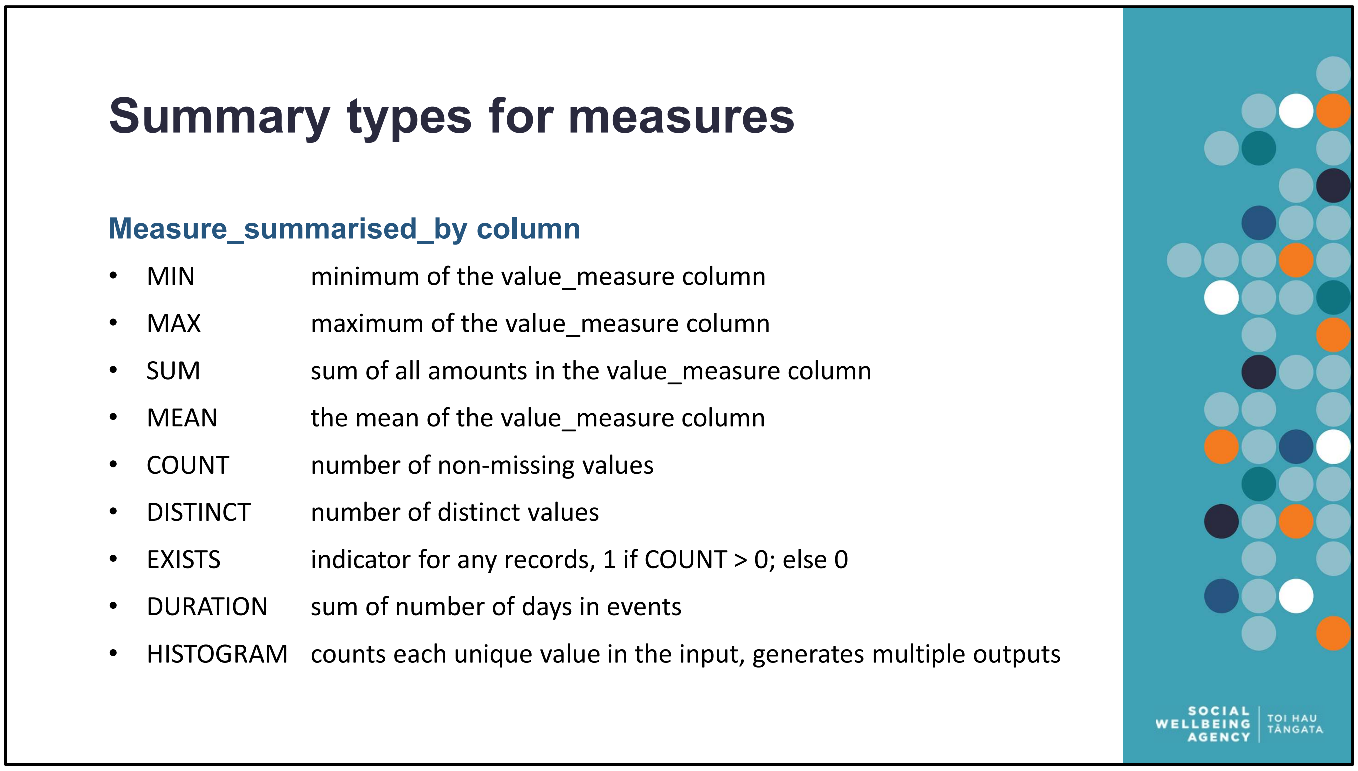
Details on each column of the population file

1. File (optional) – Recommended you record the file name of the SQL script that

creates your measure.

2. Database Name (required, sql object only) – The name of the SQL database where the measure is stored. Most likely [IDI_Clean_XXXX] or [IDI_Sandpit] or [IDI_UserCode].
3. Schema Name (required, sql object only) – The name of the schema where the measure is stored. Most likely a data-type scheme, e.g. [moe_clean], or a project schema, e.g. [DL-MAA20XX-XX].
4. Table Name (required, sql object only) – The name of the table/view where the measure is stored.
5. Identity Column (required, sql object or constant) – The name of the column that contains the unique identifier for the population. This can be an individual identifier, e.g. [snz_uid], or some other identifier, e.g. [snz_xxx_uid], [mb_2018], [school_id].
6. Measure Period Start Date (required, sql object or constant) – Start date for event. If entering constants, the data must be given as "YYYY-MM-DD" e.g. "2020-01-01" as the start of 2020.
7. Measure Period End Date (required, sql object or constant) – End date for event. If entering constants, the data must be given as "YYYY-MM-DD" e.g. "2020-12-31" as the end of 2020.
8. Label Measure (required, sql object or constant) – The label assigned to the output measure. It is recommended to avoid special characters and use underscore "_" instead of space " ".
9. Value Measure (required, sql object or constant) – The variable name that will be measured. In some cases this will be a placeholder value.
10. Measure Summarised By (required, specific list of accepted commands) – This column stores the (pre-defined) function that will be used to summarise the measure. The next slide details on what these functions are and what they do.
11. Proportional (required, TRUE or FALSE) – Proportional is used to pro rate to the intersection between the summary period and the measure period. If a variable is measured with proportional being TRUE, then the measure summary (which is usually duration / number of days) will be calculated for the period falling within the population summary start and end dates.
12. Notes / Description (optional) – Recommended you record a short description of the variable that is produced.

Value_measure may take placeholder values. For example, when calculating DURATION a column containing additional values (such as income earned) is not required. Or when counting all records, we count a placeholder "1" as counting a specific column only counts people with a non-missing value for that column. We use "1" as our placeholder.



Summary types for measures

Measure_summarised_by column

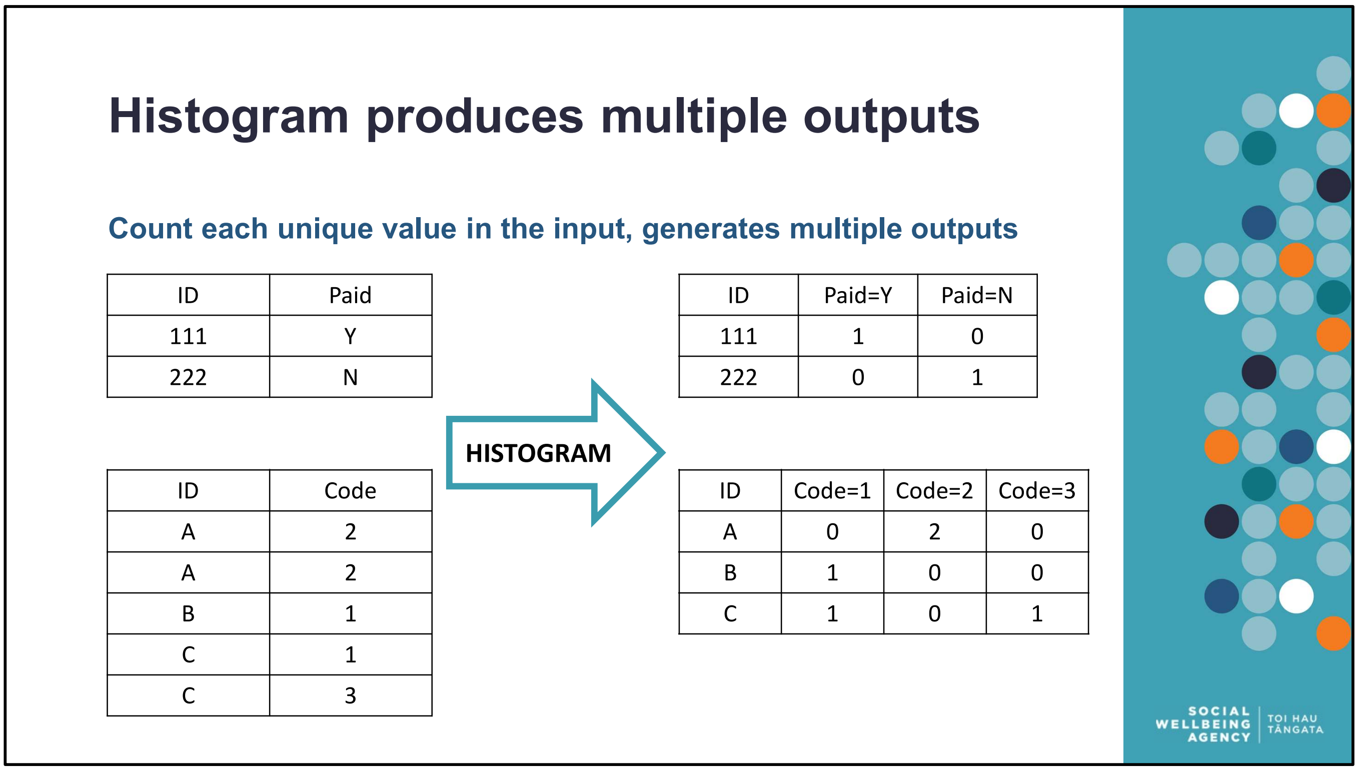
• MIN	minimum of the value_measure column
• MAX	maximum of the value_measure column
• SUM	sum of all amounts in the value_measure column
• MEAN	the mean of the value_measure column
• COUNT	number of non-missing values
• DISTINCT	number of distinct values
• EXISTS	indicator for any records, 1 if COUNT > 0; else 0
• DURATION	sum of number of days in events
• HISTOGRAM	counts each unique value in the input, generates multiple outputs

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

MIN, MAX, SUM, MEAN, DISTINCT and HISTOGRAM apply to Value_measure column in control file.
DURATION does not use Value_measure column. We use “1” as a placeholder.
COUNT and EXISTS will not count missing values in the Value_measure column. We use “1”, to count all records.

You may need to cast values into numeric type before assembling with the tool. For example, a column might contain codes “00” and “01”, but if these are stored as text then they can not be summarised using MIN, MAX, or SUM. They could be summarised using HISTOGRAM, but it is often best to convert/cast these values to numbers when creating the input measures.

Most of these functions translate direct to the equivalent SQL summary function.
MEAN can also be created by calculating SUM and COUNT and then dividing SUM by COUNT.
DISTINCT will count missing/NULL columns as a separate type (https://www.sqlservertutorial.net/sql-server-basics/sql-server-select-distinct).



Histogram produces multiple outputs

Count each unique value in the input, generates multiple outputs

ID	Paid
111	Y
222	N

HISTOGRAM

ID	Paid=Y	Paid=N
111	1	0
222	0	1

ID	Code
A	2
A	2
B	1
C	1
C	3

ID	Code=1	Code=2	Code=3
A	0	2	0
B	1	0	0
C	1	0	1

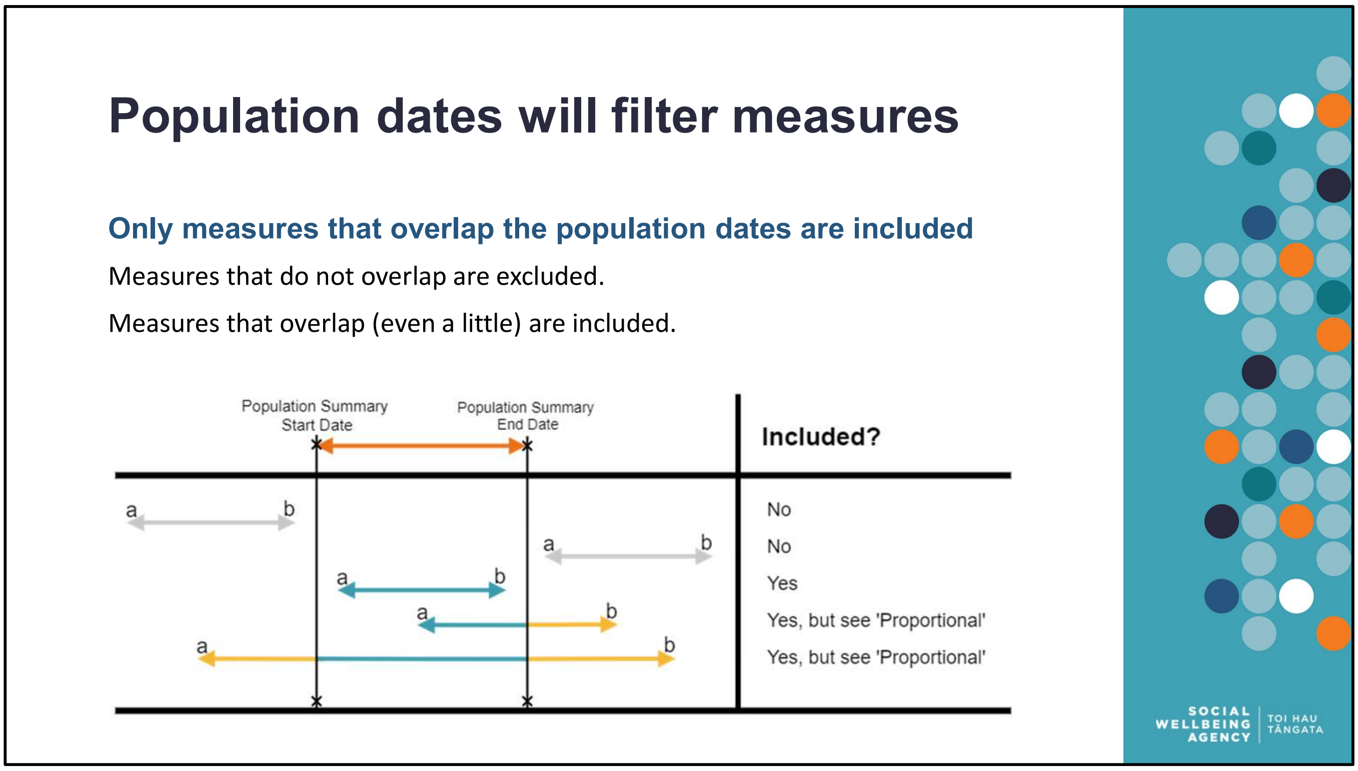
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

HISTOGRAM can be a good choice for summarising text columns that can not easily be cast into text.

HISTOGRAM can be considered as a “count each type” summary measure. It produces multiple output columns, one for each unique value in the input. This is most useful when the input column contains text codes. Because the assembly tool only outputs numeric values, HISTOGRAM can be used to produce a collection of indicator columns from text. There are helper functions in the supporting R scripts to reverse this and combine multiple columns back into a single one.

HISTOGRAM produces a new column in the output table for each unique value in the value_measure column. Before using HISTOGRAM, check that there are only as small number of unique values in the column: ~100 values (e.g. TA codes) is fine. More than 1000 unique values will produce errors because SQL has a limit of 1024 columns in a single table.

Researchers who are familiar with R may find the ‘collapse_indicator_columns’ function in dbplyr_helper_functions useful for recombining histogram columns back into a single column (where applicable).

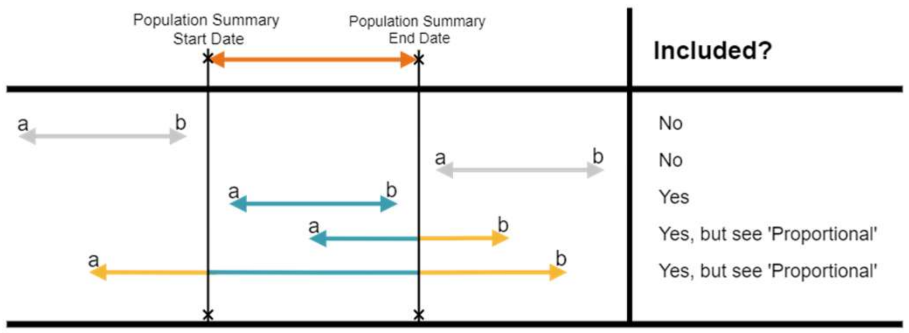


Population dates will filter measures

Only measures that overlap the population dates are included

Measures that do not overlap are excluded.

Measures that overlap (even a little) are included.



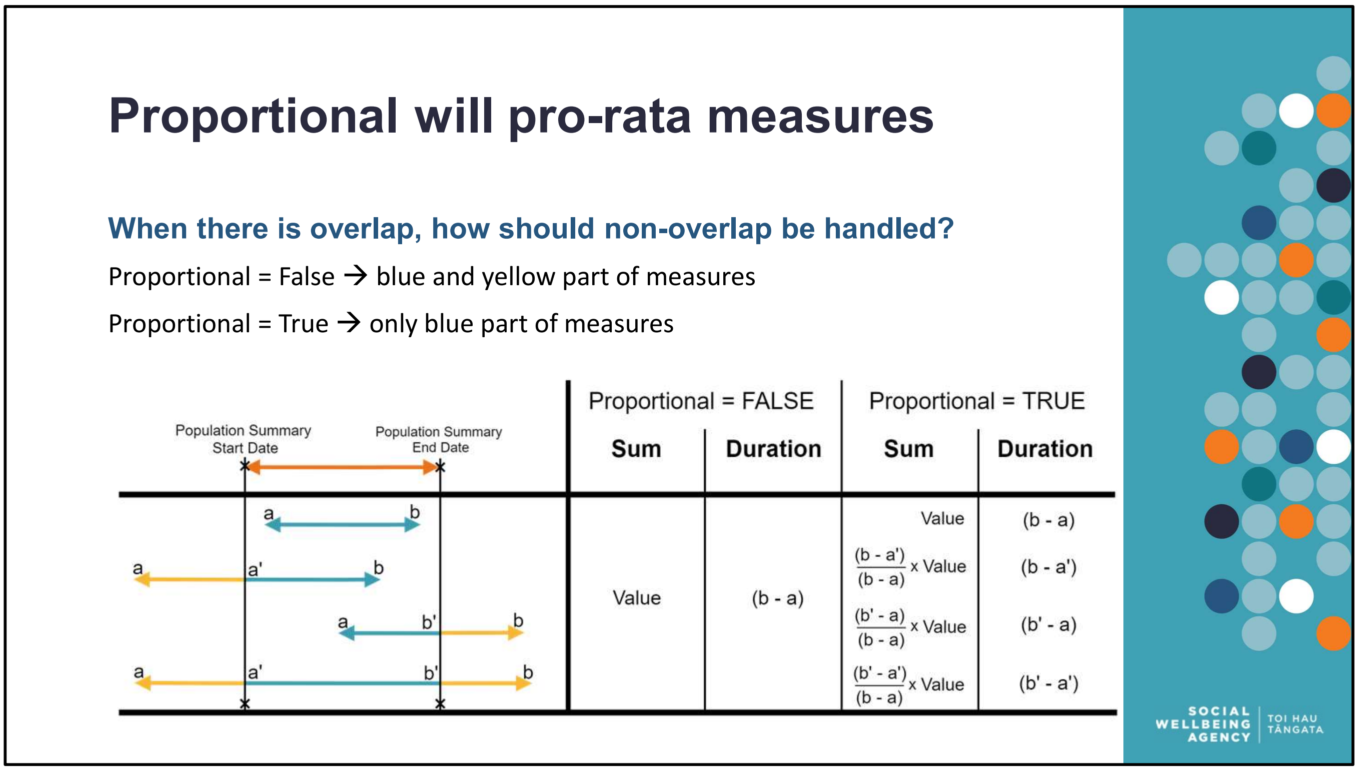
Population Summary Start Date
Population Summary End Date
Included?
a
b
No
a
b
No
a
b
Yes
a
b
Yes, but see 'Proportional'
a
b
Yes, but see 'Proportional'

This applies to all summary types. Summary dates for population are compared against dates for measures. Only measures that overlap with the summary period will be included in the output.

Orange gives the summary period for a member of the study population. Grey is excluded, Teal is included, Yellow is included but depends on the Proportional column.

In order from top to bottom in the figure:

1. Measures that end before the population summary start date are excluded.
2. Measures that start after the population summary end date are excluded.
3. Measures that sit entirely within the population summary period (measure starts after population summary starts, and measure ends before population summary ends) are included.
4. Measures that start (or end) within the population summary period but end (or start) outside the population summary period are included but are affected by the 'Proportional' column of the control file.
5. Measures that start before, and end after, the population summary period are included, but are affected by the 'Proportional' column of the control file.



Proportional will pro-rata measures

When there is overlap, how should non-overlap be handled?

Proportional = False → blue and yellow part of measures

Proportional = True → only blue part of measures

Population Summary Start Date
Population Summary End Date
a
b
a
a'
b
a
b'
b
a
a'
b'
b
SOCIAL WELLBEING AGENCY
TOI HAU TĀNGATA

	Proportional = FALSE		Proportional = TRUE	
	Sum	Duration	Sum	Duration
	Value	(b - a)	Value	(b - a)
			$\frac{(b - a')}{(b - a)}$ x Value	(b - a')
			$\frac{(b' - a)}{(b - a)}$ x Value	(b' - a)
			$\frac{(b' - a')}{(b - a)}$ x Value	(b' - a')

For DURATION, SUM, and MEAN summary types only. When there is some overlap between a measure and the summary period, Proportional determines how non-overlap is handled. Either we include the entire value/measure or we scale it down (pro-rata) to the amount that overlaps.

Orange gives the summary period for a member of the study population. Teal is included, Yellow depends on the Proportional value. The labels are:

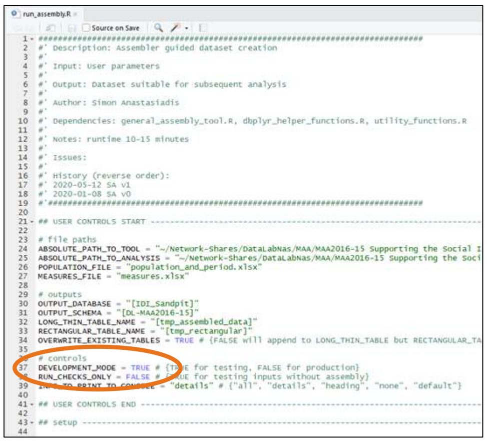
- a – the measure start date
- b – the measure end date
- a' – the population start date used when measure starts <u>before</u> the summary period.
- b' – the population end date used when measure ends <u>after</u> the summary period.

The columns to the right shows how the value of SUM or DURATION changes depending on proportional. MEAN changes in that equivalent way to SUM.

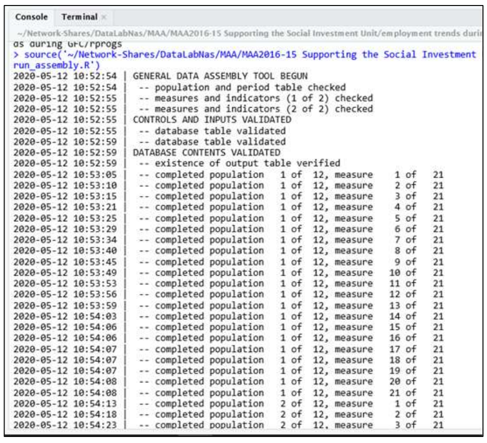
When using DURATION you will often want PROPORTIONAL = TRUE.

- PROPORTIONAL = FALSE with DURATION counts all days where the event period has some overlap with the summary period.
- PROPORTIONAL = TRUE with DURATION counts all days in both the event period and the summary period.

Setup run_assembly



```
run_assembly.R
Source on Save
################################################################################
#' Description: Assembler guided dataset creation
#'
#' Input: User parameters
#'
#' Output: Dataset suitable for subsequent analysis
#'
#' Author: Simon Anastasiadis
#'
#' Dependencies: general_assembly_tool.R, dbplyr_helper_functions.R, utility_functions.R
#'
#' Notes: runtime 10-15 minutes
#'
#' Issues:
#'
#' History (reverse order):
#' 2020-05-12 SA v1
#' 2020-01-08 SA v0
#'###############################################################################

## USER CONTROLS START ------------------------------------------------------------

# file paths
ABSOLUTE_PATH_TO_TOOL = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social I
ABSOLUTE_PATH_TO_ANALYSIS = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Soci
POPULATION_FILE = "population_and_period.xlsx"
MEASURES_FILE = "measures.xlsx"

# outputs
OUTPUT_DATABASE = "[IDI_Sandpit]"
OUTPUT_SCHEMA = "[DL-MAA2016-15]"
LONG_THIN_TABLE_NAME = "[tmp_assembled_data]"
RECTANGULAR_TABLE_NAME = "[tmp_rectangular]"
OVERWRITE_EXISTING_TABLES = TRUE # {FALSE will append to LONG_THIN_TABLE but RECTANGULAR_TA

# controls
DEVELOPMENT_MODE = TRUE # {TRUE for testing, FALSE for production}
RUN_CHECKS_ONLY = FALSE # {TRUE for testing inputs without assembly}
INFO_TO_PRINT_TO_CONSOLE = "details" # {"all", "details", "heading", "none", "default"}

## USER CONTROLS END --------------------------------------------------------------

## setup --------------------------------------------------------------------------

```



```
Console   Terminal
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends duri
as during GFC/rprogs
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment
run_assembly.R')
2020-05-12 10:52:54 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 10:52:54 |  -- population and period table checked
2020-05-12 10:52:55 |  -- measures and indicators (1 of 2) checked
2020-05-12 10:52:55 |  -- measures and indicators (2 of 2) checked
2020-05-12 10:52:55 | CONTROLS AND INPUTS VALIDATED
2020-05-12 10:52:55 |  -- database table validated
2020-05-12 10:52:59 |  -- database table validated
2020-05-12 10:52:59 | DATABASE CONTENTS VALIDATED
2020-05-12 10:52:59 |  -- existence of output table verified
2020-05-12 10:53:05 |  -- completed population  1 of  12, measure   1 of   21
2020-05-12 10:53:10 |  -- completed population  1 of  12, measure   2 of   21
2020-05-12 10:53:15 |  -- completed population  1 of  12, measure   3 of   21
2020-05-12 10:53:21 |  -- completed population  1 of  12, measure   4 of   21
2020-05-12 10:53:25 |  -- completed population  1 of  12, measure   5 of   21
2020-05-12 10:53:29 |  -- completed population  1 of  12, measure   6 of   21
2020-05-12 10:53:34 |  -- completed population  1 of  12, measure   7 of   21
2020-05-12 10:53:40 |  -- completed population  1 of  12, measure   8 of   21
2020-05-12 10:53:45 |  -- completed population  1 of  12, measure   9 of   21
2020-05-12 10:53:49 |  -- completed population  1 of  12, measure  10 of   21
2020-05-12 10:53:53 |  -- completed population  1 of  12, measure  11 of   21
2020-05-12 10:53:56 |  -- completed population  1 of  12, measure  12 of   21
2020-05-12 10:53:59 |  -- completed population  1 of  12, measure  13 of   21
2020-05-12 10:54:03 |  -- completed population  1 of  12, measure  14 of   21
2020-05-12 10:54:06 |  -- completed population  1 of  12, measure  15 of   21
2020-05-12 10:54:06 |  -- completed population  1 of  12, measure  16 of   21
2020-05-12 10:54:07 |  -- completed population  1 of  12, measure  17 of   21
2020-05-12 10:54:07 |  -- completed population  1 of  12, measure  18 of   21
2020-05-12 10:54:07 |  -- completed population  1 of  12, measure  19 of   21
2020-05-12 10:54:08 |  -- completed population  1 of  12, measure  20 of   21
2020-05-12 10:54:08 |  -- completed population  1 of  12, measure  21 of   21
2020-05-12 10:54:13 |  -- completed population  2 of  12, measure   1 of   21
2020-05-12 10:54:18 |  -- completed population  2 of  12, measure   2 of   21
2020-05-12 10:54:23 |  -- completed population  2 of  12, measure   3 of   21
```

While you can call the assembly tool programmatically from R, the script run_assembly.R has been created to simplify the most common case: assembling a rectangular dataset from a pair of control files.

Enter the details in the section labelled user controls.
For your first run, ensure that development mode is set to TRUE.

You can then run the script
- click the bottom in the top-right of the script window labelled "Source"
- OR press Ctrl + Shift + S

You will start to see console output (info messages for the user) like on the right.
Next slide shows the end of the console output.

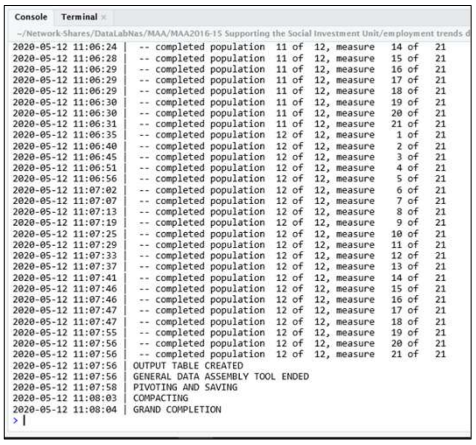
Note the tool (attempts to) assembles every measure (row in the measures table) for every population (row in the populations table). Hence in the example above, the tool covers 21 measures for 12 populations (it was a panel of 12 quarters).

Trial in development mode

Early check

Development model limits processing to only the top 1000 records

- Test on a subset of the data
- Find errors quickly
- Check for measures performing especially slowly



Console Terminal

~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends d

```
2020-05-12 11:06:24 |  -- completed population  11 of  12, measure   14 of   21
2020-05-12 11:06:28 |  -- completed population  11 of  12, measure   15 of   21
2020-05-12 11:06:29 |  -- completed population  11 of  12, measure   16 of   21
2020-05-12 11:06:29 |  -- completed population  11 of  12, measure   17 of   21
2020-05-12 11:06:29 |  -- completed population  11 of  12, measure   18 of   21
2020-05-12 11:06:30 |  -- completed population  11 of  12, measure   19 of   21
2020-05-12 11:06:30 |  -- completed population  11 of  12, measure   20 of   21
2020-05-12 11:06:31 |  -- completed population  11 of  12, measure   21 of   21
2020-05-12 11:06:35 |  -- completed population  12 of  12, measure    1 of   21
2020-05-12 11:06:40 |  -- completed population  12 of  12, measure    2 of   21
2020-05-12 11:06:45 |  -- completed population  12 of  12, measure    3 of   21
2020-05-12 11:06:51 |  -- completed population  12 of  12, measure    4 of   21
2020-05-12 11:06:56 |  -- completed population  12 of  12, measure    5 of   21
2020-05-12 11:07:02 |  -- completed population  12 of  12, measure    6 of   21
2020-05-12 11:07:07 |  -- completed population  12 of  12, measure    7 of   21
2020-05-12 11:07:13 |  -- completed population  12 of  12, measure    8 of   21
2020-05-12 11:07:19 |  -- completed population  12 of  12, measure    9 of   21
2020-05-12 11:07:25 |  -- completed population  12 of  12, measure   10 of   21
2020-05-12 11:07:29 |  -- completed population  12 of  12, measure   11 of   21
2020-05-12 11:07:33 |  -- completed population  12 of  12, measure   12 of   21
2020-05-12 11:07:37 |  -- completed population  12 of  12, measure   13 of   21
2020-05-12 11:07:41 |  -- completed population  12 of  12, measure   14 of   21
2020-05-12 11:07:46 |  -- completed population  12 of  12, measure   15 of   21
2020-05-12 11:07:46 |  -- completed population  12 of  12, measure   16 of   21
2020-05-12 11:07:47 |  -- completed population  12 of  12, measure   17 of   21
2020-05-12 11:07:47 |  -- completed population  12 of  12, measure   18 of   21
2020-05-12 11:07:55 |  -- completed population  12 of  12, measure   19 of   21
2020-05-12 11:07:56 |  -- completed population  12 of  12, measure   20 of   21
2020-05-12 11:07:56 |  -- completed population  12 of  12, measure   21 of   21
2020-05-12 11:07:56 | OUTPUT TABLE CREATED
2020-05-12 11:07:56 | GENERAL DATA ASSEMBLY TOOL ENDED
2020-05-12 11:07:58 | PIVOTING AND SAVING
2020-05-12 11:08:03 | COMPACTING
2020-05-12 11:08:04 | GRAND COMPLETION
> |
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Image gives the end of the console output (info messages for the user) from a run. Once development mode is finishes, it is good practice to inspect the output table in SQL to ensure everything is as expected.

Why development mode? This is a design pattern I use to avoid long wait times during development/testing. Because large datasets can require large amounts of processing time DEVELOPMENT_MODE = TRUE limits the program to only the top 1000 records from each measure. This helps check that every input dataset performs as expected, but not so large that we wait several hours for a failure. Ideally the tool will finish in <10 minutes in development mode.

A good thing to check from the console log is how long did each measure take to run. Measures that are slow to run may need improvement (more efficient SQL code, convert views to tables, add indexes to tables).

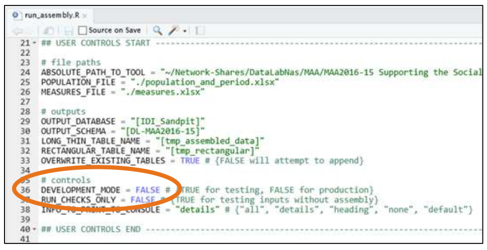
Even if development mode runs without error or concern this does not guarantee producing the full dataset will be error free. We have observed cases when no error occurs with development mode turned on but does occur with development mode turned off, because the record that causes the error is not in the first 1000 records.

Production run

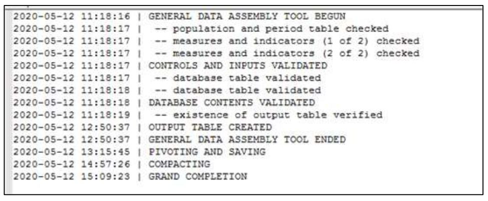
Produce full dataset

Two outputs

- Long-thin table, one row per measure and identity
- Rectangular table, one row per identity, one column per measure



```
run_assembly.R
## USER CONTROLS START ----------------------------------------------------

# file paths
ABSOLUTE_PATH_TO_TOOL = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social
POPULATION_FILE = "./population_and_period.xlsx"
MEASURES_FILE = "./measures.xlsx"

# outputs
OUTPUT_DATABASE = "[IDI_Sandpit]"
OUTPUT_SCHEMA = "[DL-MAA2016-15]"
LONG_THIN_TABLE_NAME = "[tmp_assembled_data]"
RECTANGULAR_TABLE_NAME = "[tmp_rectangular]"
OVERWRITE_EXISTING_TABLES = TRUE # {FALSE will attempt to append}

# controls
DEVELOPMENT_MODE = FALSE # {TRUE for testing, FALSE for production}
RUN_CHECKS_ONLY = FALSE # {TRUE for testing inputs without assembly}
INFO_TO_PRINT_TO_CONSOLE = "details" # {"all", "details", "heading", "none", "default"}

## USER CONTROLS END ------------------------------------------------------
```



```
2020-05-12 11:18:16 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 11:18:17 |  -- population and period table checked
2020-05-12 11:18:17 |  -- measures and indicators (1 of 2) checked
2020-05-12 11:18:17 |  -- measures and indicators (2 of 2) checked
2020-05-12 11:18:17 | CONTROLS AND INPUTS VALIDATED
2020-05-12 11:18:17 |  -- database table validated
2020-05-12 11:18:18 |  -- database table validated
2020-05-12 11:18:18 | DATABASE CONTENTS VALIDATED
2020-05-12 11:18:19 |  -- existence of output table verified
2020-05-12 12:50:37 | OUTPUT TABLE CREATED
2020-05-12 12:50:37 | GENERAL DATA ASSEMBLY TOOL ENDED
2020-05-12 13:15:45 | PIVOTING AND SAVING
2020-05-12 14:57:26 | COMPACTING
2020-05-12 15:09:23 | GRAND COMPLETION
```

How long does it take?

5 million people

21 measures

12 time periods

Took:

4 hours

20 minutes per time period

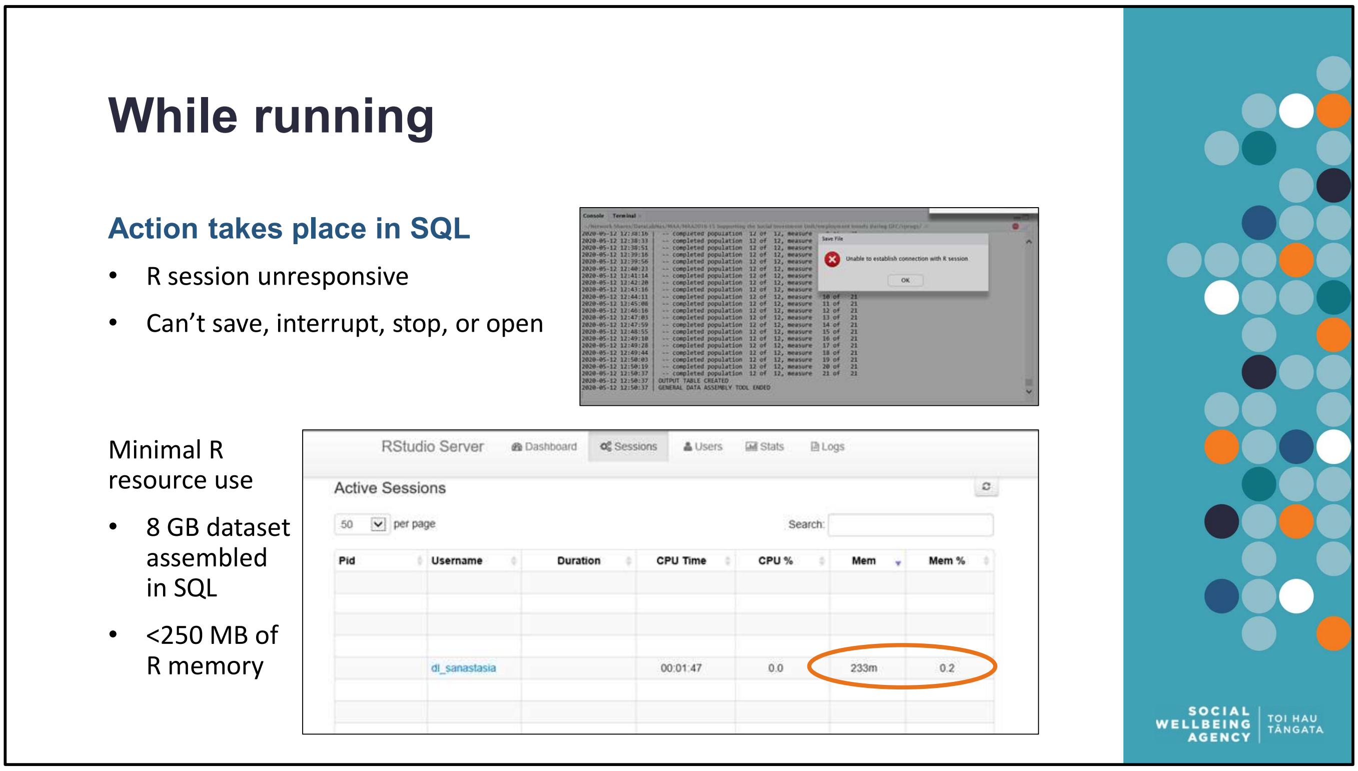
Once you have tested using development mode, then you are ready to produce your full dataset.
Setting DEVELOPMENT_MODE = FALSE means the tool will assemble the entire dataset.
This will take longer than development mode.

RHS gives example of a large production run.
For 5 million NZers, 21 measures, and 12 time periods: Total time taken was 4 hours.
This is ~20 minutes per time period (for a single cross-section).

So, an entire panel dataset assembled in a day (but don't sit waiting for it to finish).
And, a cross-sectional dataset assembled while you plan the next steps.

run_assembly.R requires you to specify two output table names.
- A long-thin table directly produced by the tool. It is easiest for the tool to produce one row per identity-and-measure-pair.
- A rectangular table for researchers with one row per identity and one column per measure.

For a worked example see the outline document.



While running

Action takes place in SQL

- R session unresponsive
- Can’t save, interrupt, stop, or open

```
2020-05-12 12:38:16 |  -- completed population  12 of  12, measure
2020-05-12 12:38:33 |  -- completed population  12 of  12, measure
2020-05-12 12:38:51 |  -- completed population  12 of  12, measure
2020-05-12 12:39:16 |  -- completed population  12 of  12, measure
2020-05-12 12:39:56 |  -- completed population  12 of  12, measure
2020-05-12 12:40:23 |  -- completed population  12 of  12, measure
2020-05-12 12:41:14 |  -- completed population  12 of  12, measure
2020-05-12 12:42:20 |  -- completed population  12 of  12, measure
2020-05-12 12:43:16 |  -- completed population  12 of  12, measure
2020-05-12 12:44:11 |  -- completed population  12 of  12, measure   10 of   21
2020-05-12 12:45:08 |  -- completed population  12 of  12, measure   11 of   21
2020-05-12 12:46:16 |  -- completed population  12 of  12, measure   12 of   21
2020-05-12 12:47:03 |  -- completed population  12 of  12, measure   13 of   21
2020-05-12 12:47:59 |  -- completed population  12 of  12, measure   14 of   21
2020-05-12 12:48:55 |  -- completed population  12 of  12, measure   15 of   21
2020-05-12 12:49:10 |  -- completed population  12 of  12, measure   16 of   21
2020-05-12 12:49:28 |  -- completed population  12 of  12, measure   17 of   21
2020-05-12 12:49:44 |  -- completed population  12 of  12, measure   18 of   21
2020-05-12 12:50:03 |  -- completed population  12 of  12, measure   19 of   21
2020-05-12 12:50:19 |  -- completed population  12 of  12, measure   20 of   21
2020-05-12 12:50:37 |  -- completed population  12 of  12, measure   21 of   21
2020-05-12 12:50:37 | OUTPUT TABLE CREATED
2020-05-12 12:50:37 | GENERAL DATA ASSEMBLY TOOL ENDED
```

Save File

Unable to establish connection with R session

OK

Minimal R resource use

- 8 GB dataset assembled in SQL
- <250 MB of R memory

Active Sessions

50 per page

Search:

Pid	Username	Duration	CPU Time	CPU %	Mem	Mem %
	dl_sanastasia		00:01:47	0.0	233m	0.2

Assemble the entire dataset takes longer than development mode. Perhaps 10 times longer.

Because most of the assembly takes place on the SQL server, the tool puts minimal pressure on R memory or resources.
However, while it is running the tool is making consistent use of SQL server resources. So you still need to be considerate of other users – the datalab is a shared resource.

Even though SQL server does most of the work, your R session will still be unresponsive while the tool is running as R waits for SQL.
If you are an R developer, you may want to run two R sessions – one for the tool and the other for development.

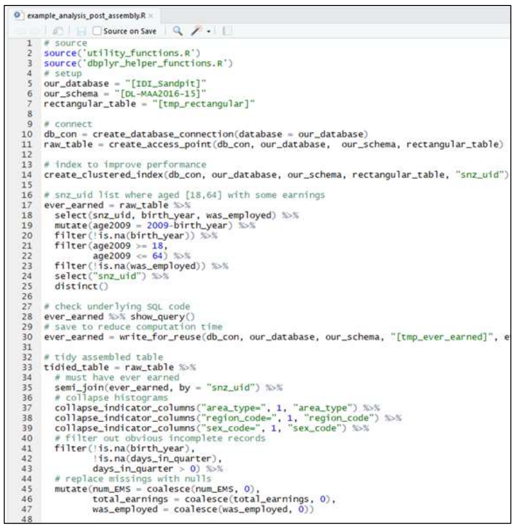
- You will need to open the second session before starting the tool.
- You are responsible for closing both R sessions – the datalab is a shared resource.

The only way I have found to stop R running when it is waiting for the SQL server to respond is to close the browser (R Studio is accessed in the IDI via a web-browser), reopen it, and log back in. You will then be prompted that R is non-responsive do you want to stop any running process.

Conduct your analysis

Begin by tidying up dataset

- Replace missing values with zero
- Collapse histogram columns to indicators
- Checking measures for outliers
- Explore and visualise
- Etc.



example_analysis_post_assembly.R

```r
# source
source('utility_functions.R')
source('dbplyr_helper_functions.R')
# setup
our_database = "[IDI_Sandpit]"
our_schema = "[DL-MAA2016-15]"
rectangular_table = "[tmp_rectangular]"

# connect
db_con = create_database_connection(database = our_database)
raw_table = create_access_point(db_con, our_database,  our_schema, rectangular_table)

# index to improve performance
create_clustered_index(db_con, our_database, our_schema, rectangular_table, "snz_uid")

# snz_uid list where aged [18,64] with some earnings
ever_earned = raw_table %>%
  select(snz_uid, birth_year, was_employed) %>%
  mutate(age2009 = 2009-birth_year) %>%
  filter(!is.na(birth_year)) %>%
  filter(age2009 >= 18,
         age2009 <= 64) %>%
  filter(!is.na(was_employed)) %>%
  select("snz_uid") %>%
  distinct()

# check underlying SQL code
ever_earned %>% show_query()
# save to reduce computation time
ever_earned = write_for_reuse(db_con, our_database, our_schema, "[tmp_ever_earned]", e

# tidy assembled table
tidied_table = raw_table %>%
  # must have ever earned
  semi_join(ever_earned, by = "snz_uid") %>%
  # collapse histograms
  collapse_indicator_columns("area_type=", 1, "area_type") %>%
  collapse_indicator_columns("region_code=", 1, "region_code") %>%
  collapse_indicator_columns("sex_code=", 1, "sex_code") %>%
  # filter out obvious incomplete records
  filter(!is.na(birth_year),
         !is.na(days_in_quarter),
         days_in_quarter > 0) %>%
  # replace missings with nulls
  mutate(num_EMS = coalesce(num_EMS, 0),
         total_earnings = coalesce(total_earnings, 0),
         was_employed = coalesce(was_employed, 0))

```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

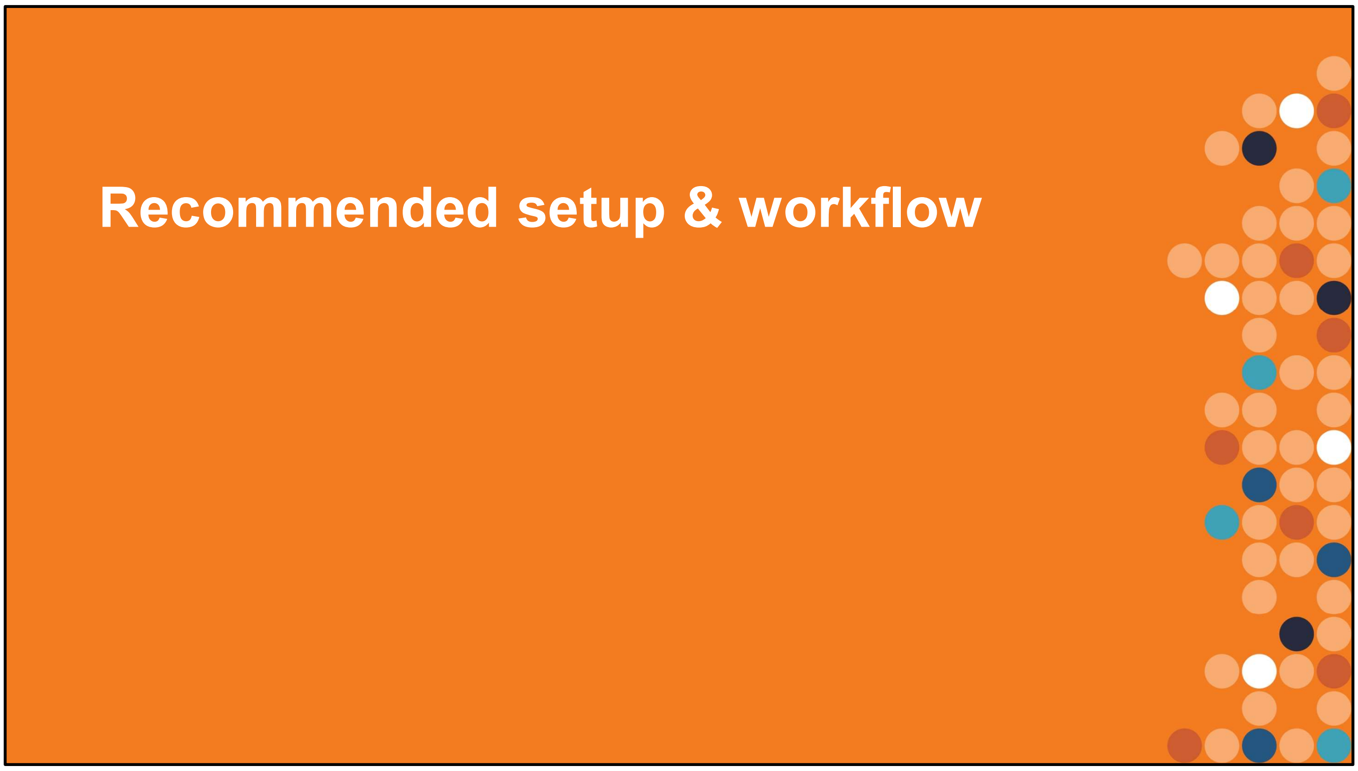
This tool is an assembly tool. Unless you conducted thorough data cleaning before assembly, you will need to conduct it afterwards.

Two most common tasks post-assembly tool are:

- Replace missing values with zeros: If a measure is sum income and a person has no income, then this value will be missing/NULL for them. You will need to decide how to handle missing values. In many cases you will want to replace missing values with zeros.
- Collapse histogram columns: The histogram summary command produces multiple columns, one for every distinct value in the input column. This is necessary because the assembly tool requires numeric outputs as certain stages. These can be collapse back from a series of indicator columns to a single column.

For data exploration and visualisation, I recommend

1. Filtering to at most 50,000 rows
2. Collecting the data into R
3. Visualising it with the explore package (https://cran.r-project.org/web/packages/explore/vignettes/explore_mtcars.html)



Recommended setup & workflow

While you are welcome to use the assembly tool in the way that best suits you (being a good citizen of the shared datalab resources), I have found it to be especially effective as part of a broader workflow pattern.

Consider adopting one or more of the following practices to improve your IDI research.

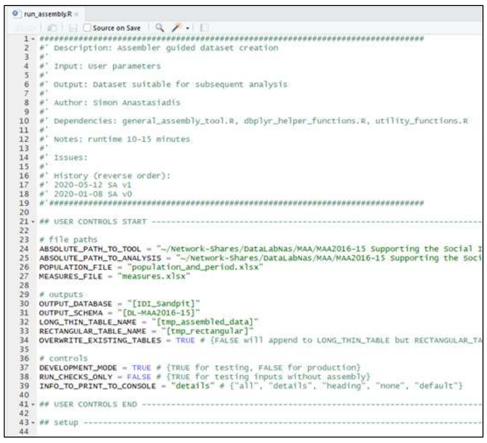
Only one copy of the tool

Each analysis needs its own

- Assembly file
- Excel population control file
- Excel measure control file

But it does not need its own copy of the assembly tool

Reduce clutter by using a single copy


run_assembly.R

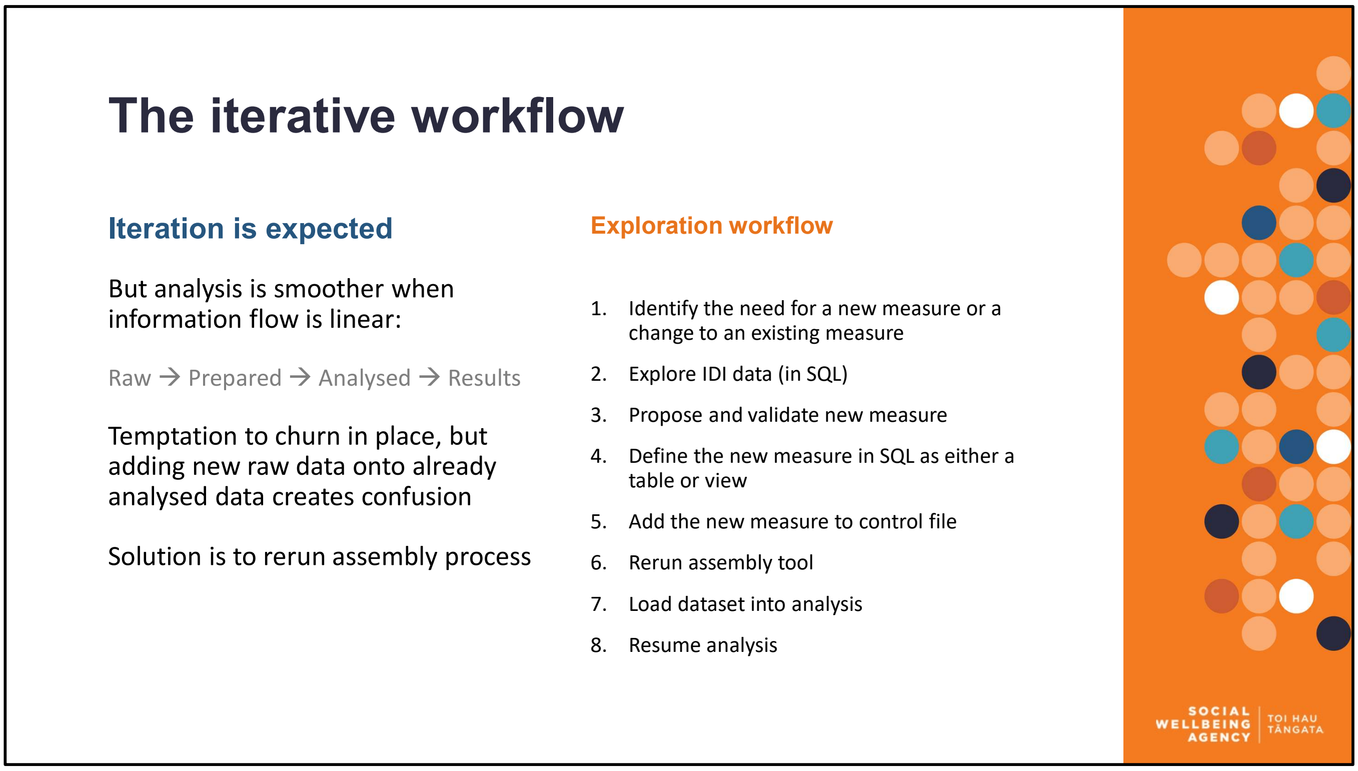
```
################################################################################
#' Description: Assembler guided dataset creation
#'
#' Input: User parameters
#'
#' Output: Dataset suitable for subsequent analysis
#'
#' Author: Simon Anastasiadis
#'
#' Dependencies: general_assembly_tool.R, dbplyr_helper_functions.R, utility_functions.R
#'
#' Notes: runtime 10-15 minutes
#'
#' Issues:
#'
#' History (reverse order):
#' 2020-05-12 SA v1
#' 2020-01-08 SA v0
#'###############################################################################

## USER CONTROLS START -------------------------------------------------------------

# file paths
ABSOLUTE_PATH_TO_TOOL = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social I
ABSOLUTE_PATH_TO_ANALYSIS = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Soci
POPULATION_FILE = "population_and_period.xlsx"
MEASURES_FILE = "measures.xlsx"

# outputs
OUTPUT_DATABASE = "[IDI_Sandpit]"
OUTPUT_SCHEMA = "[DL-MAA2016-15]"
LONG_THIN_TABLE_NAME = "[tmp_assembled_data]"
RECTANGULAR_TABLE_NAME = "[tmp_rectangular]"
OVERWRITE_EXISTING_TABLES = TRUE # {FALSE will append to LONG_THIN_TABLE but RECTANGULAR_TA

# controls
DEVELOPMENT_MODE = TRUE # {TRUE for testing, FALSE for production}
RUN_CHECKS_ONLY = FALSE # {TRUE for testing inputs without assembly}
INFO_TO_PRINT_TO_CONSOLE = "details" # {"all", "details", "heading", "none", "default"}

## USER CONTROLS END ---------------------------------------------------------------

## setup ---------------------------------------------------------------------------

```

While it can be tempting to have a copy of the Dataset Assembly Tool for each separate piece of analysis, only one copy of the tool is needed for an entire research project.

run_assembly.R requires two paths: the location of the tool and the location of the analysis.

- The location of the tool is the folder where general_assembly_tool.R and automated_tests.R are stored
- The location of the analysis is where run_assembly.R and the Excel control files are stored

When you run the assembly tool, it will first load the tool from the first folder and then carry out the assembly specified in the second folder.
Therefore you can have many Excel control files and many different datasets assembled, all using the same copy of the tool.



The iterative workflow

Iteration is expected

But analysis is smoother when information flow is linear:

Raw → Prepared → Analysed → Results

Temptation to churn in place, but adding new raw data onto already analysed data creates confusion

Solution is to rerun assembly process

Exploration workflow

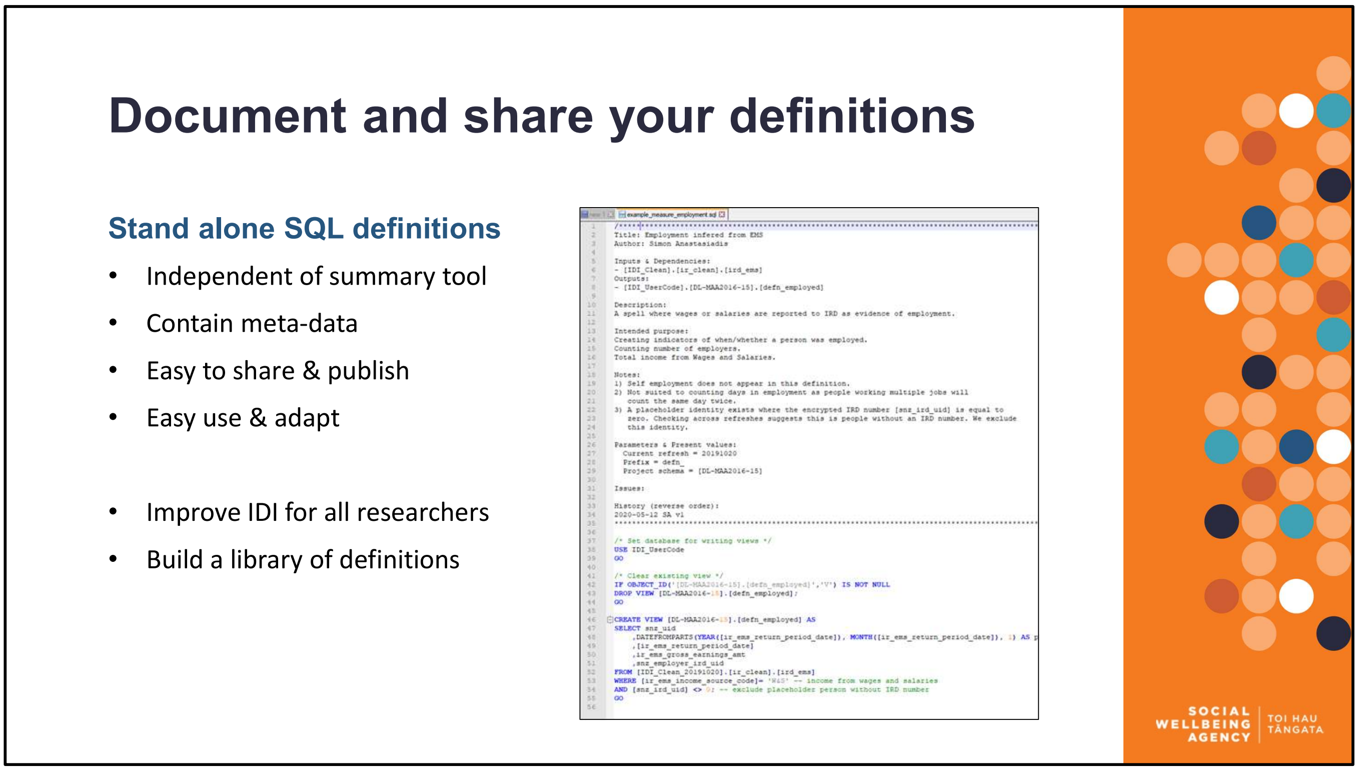
1. Identify the need for a new measure or a change to an existing measure
2. Explore IDI data (in SQL)
3. Propose and validate new measure
4. Define the new measure in SQL as either a table or view
5. Add the new measure to control file
6. Rerun assembly tool
7. Load dataset into analysis
8. Resume analysis

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

The nature of research and development is that it is non-linear: Data preparation and preliminary analysis reveals that other data is needed, so additional data is gathered, further results are produced, and the process iterates. While your process may be iterative and non-linear, the flow of information through your analysis can be linear.

Having a linear flow of information simplifies and clarifies your analysis. The goal is that at each stage all the data is in the same state:
Raw unformed IDI → prepared events → freshly assembled data → cleaned and analysis ready → results
In this way a single dataset is never a mixture of both raw & analysed data. Don't add raw data to analysed data, add raw data to other raw data and analyse it all together.

Because the assembly tool is designed to be rerun, it helps ensure that the initial part of your data process is linear. Designing your data cleaning and analysis to be repeated once new data is added or definitions are changed, helps ensure you maintain this linear process.



Document and share your definitions

Stand alone SQL definitions

- Independent of summary tool
- Contain meta-data
- Easy to share & publish
- Easy use & adapt

- Improve IDI for all researchers
- Build a library of definitions

example_measure_employment.sql

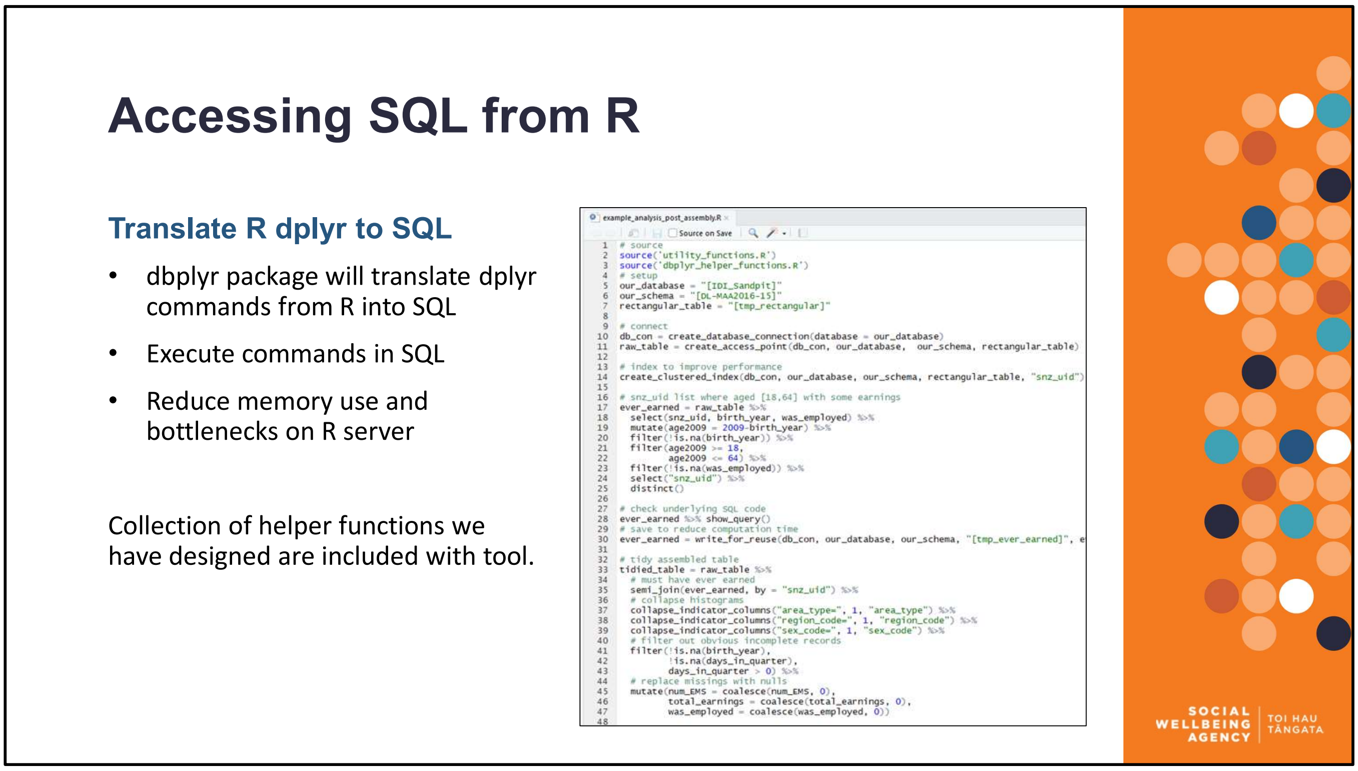
```
/*****************************************************************************
Title: Employment infered from EMS
Author: Simon Anastasiadis

Inputs & Dependencies:
- [IDI_Clean].[ir_clean].[ird_ems]
Outputs:
- [IDI_UserCode].[DL-MAA2016-15].[defn_employed]

Description:
A spell where wages or salaries are reported to IRD as evidence of employment.

Intended purpose:
Creating indicators of when/whether a person was employed.
Counting number of employers.
Total income from Wages and Salaries.

Notes:
1) Self employment does not appear in this definition.
2) Not suited to counting days in employment as people working multiple jobs will
   count the same day twice.
3) A placeholder identity exists where the encrypted IRD number [snz_ird_uid] is equal to
   zero. Checking across refreshes suggests this is people without an IRD number. We exclude
   this identity.

Parameters & Present values:
  Current refresh = 20191020
  Prefix = defn_
  Project schema = [DL-MAA2016-15]

Issues:

History (reverse order):
2020-05-12 SA v1
*****************************************************************************

/* Set database for writing views */
USE IDI_UserCode
GO

/* Clear existing view */
IF OBJECT_ID('[DL-MAA2016-15].[defn_employed]','V') IS NOT NULL
DROP VIEW [DL-MAA2016-15].[defn_employed];
GO

CREATE VIEW [DL-MAA2016-15].[defn_employed] AS
SELECT snz_uid
	,DATEFROMPARTS(YEAR([ir_ems_return_period_date]), MONTH([ir_ems_return_period_date]), 1) AS p
	,[ir_ems_return_period_date]
	,ir_ems_gross_earnings_amt
	,snz_employer_ird_uid
FROM [IDI_Clean_20191020].[ir_clean].[ird_ems]
WHERE [ir_ems_income_source_code]= 'W&S' -- income from wages and salaries
AND [snz_ird_uid] <> 0; -- exclude placeholder person without IRD number
GO
```

Once you have researched the raw data, constructed a definition for a specific measure, and validated the correctness of the measure it is good practice to document your work. Writing up some documentation will be less effort than having to retrace your steps if you forget, and should take less time than the tool saves you.

Sharing your definitions is part of publishing your code. As more of us do this, we will together build a Library of Definitions. This will be a significant step in the maturity of the IDI. Imagine being able to get health definitions created by health experts, or benefit definitions created by benefit experts. The breadth of the IDI means we need a wide range of subject matter experts – this is a practical way to start.

I propose these definitions are written in SQL. This makes them accessible to both SAS and R users. Even if they prefer R or SAS, most IDI researchers know a little SQL anyway because that is how you view the raw data.

The ‘expanded header’ is the current documentation approach. Not necessarily an excellent approach, but more important to get started with an approach than to ensure the approach is perfect before starting. As the library of definitions grows we can expect this process will be refined.



Accessing SQL from R

Translate R dplyr to SQL

- dbplyr package will translate dplyr commands from R into SQL
- Execute commands in SQL
- Reduce memory use and bottlenecks on R server

Collection of helper functions we have designed are included with tool.

example_analysis_post_assembly.R

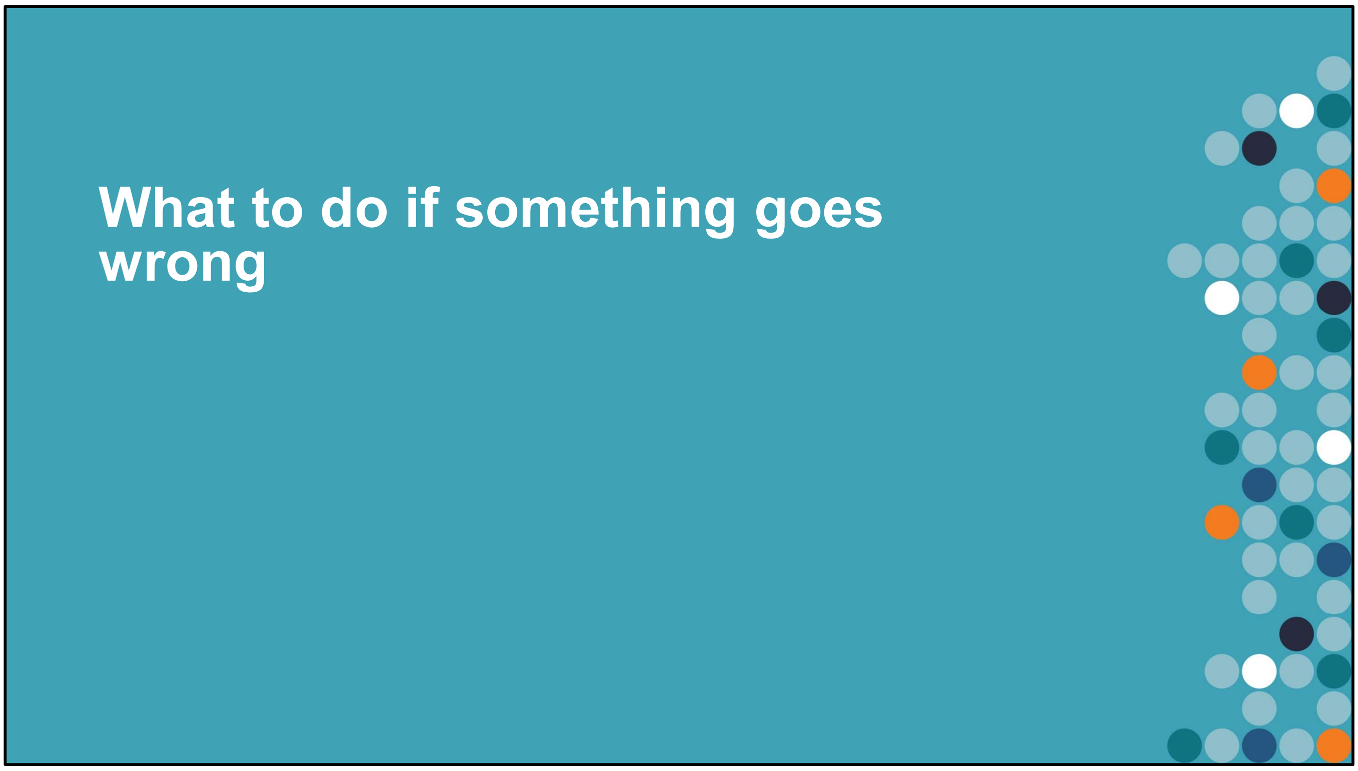
```r
# source
source('utility_functions.R')
source('dbplyr_helper_functions.R')
# setup
our_database = "[IDI_Sandpit]"
our_schema = "[DL-MAA2016-15]"
rectangular_table = "[tmp_rectangular]"

# connect
db_con = create_database_connection(database = our_database)
raw_table = create_access_point(db_con, our_database,  our_schema, rectangular_table)

# index to improve performance
create_clustered_index(db_con, our_database, our_schema, rectangular_table, "snz_uid")

# snz_uid list where aged [18,64] with some earnings
ever_earned = raw_table %>%
  select(snz_uid, birth_year, was_employed) %>%
  mutate(age2009 = 2009-birth_year) %>%
  filter(!is.na(birth_year)) %>%
  filter(age2009 >= 18,
         age2009 <= 64) %>%
  filter(!is.na(was_employed)) %>%
  select("snz_uid") %>%
  distinct()

# check underlying SQL code
ever_earned %>% show_query()
# save to reduce computation time
ever_earned = write_for_reuse(db_con, our_database, our_schema, "[tmp_ever_earned]", e

# tidy assembled table
tidied_table = raw_table %>%
  # must have ever earned
  semi_join(ever_earned, by = "snz_uid") %>%
  # collapse histograms
  collapse_indicator_columns("area_type=", 1, "area_type") %>%
  collapse_indicator_columns("region_code=", 1, "region_code") %>%
  collapse_indicator_columns("sex_code=", 1, "sex_code") %>%
  # filter out obvious incomplete records
  filter(!is.na(birth_year),
         !is.na(days_in_quarter),
         days_in_quarter > 0) %>%
  # replace missings with nulls
  mutate(num_EMS = coalesce(num_EMS, 0),
         total_earnings = coalesce(total_earnings, 0),
         was_employed = coalesce(was_employed, 0))
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

While SQL is well suited to defining populations, events, and measures SAS and R tend to be preferred for programming.
Just as it is possible to connect SAS to SQL, write SAS commands, and have them executed by SQL,
it is also possible to connect R to SQL, write R commands, and have them executed by SQL.

This reduces memory use and bottlenecks on the R server. Uses SQL for its strength – working across large datasets. Collect into R once prepared and tidied to use R for its strength – statistical analysis.

The key R package for doing this is dbplyr. Along with the assembly tool, we also provide a collection of helper functions to support researchers wanting to take this approach. An example using this approach is included in the documentation file (displayed in image).

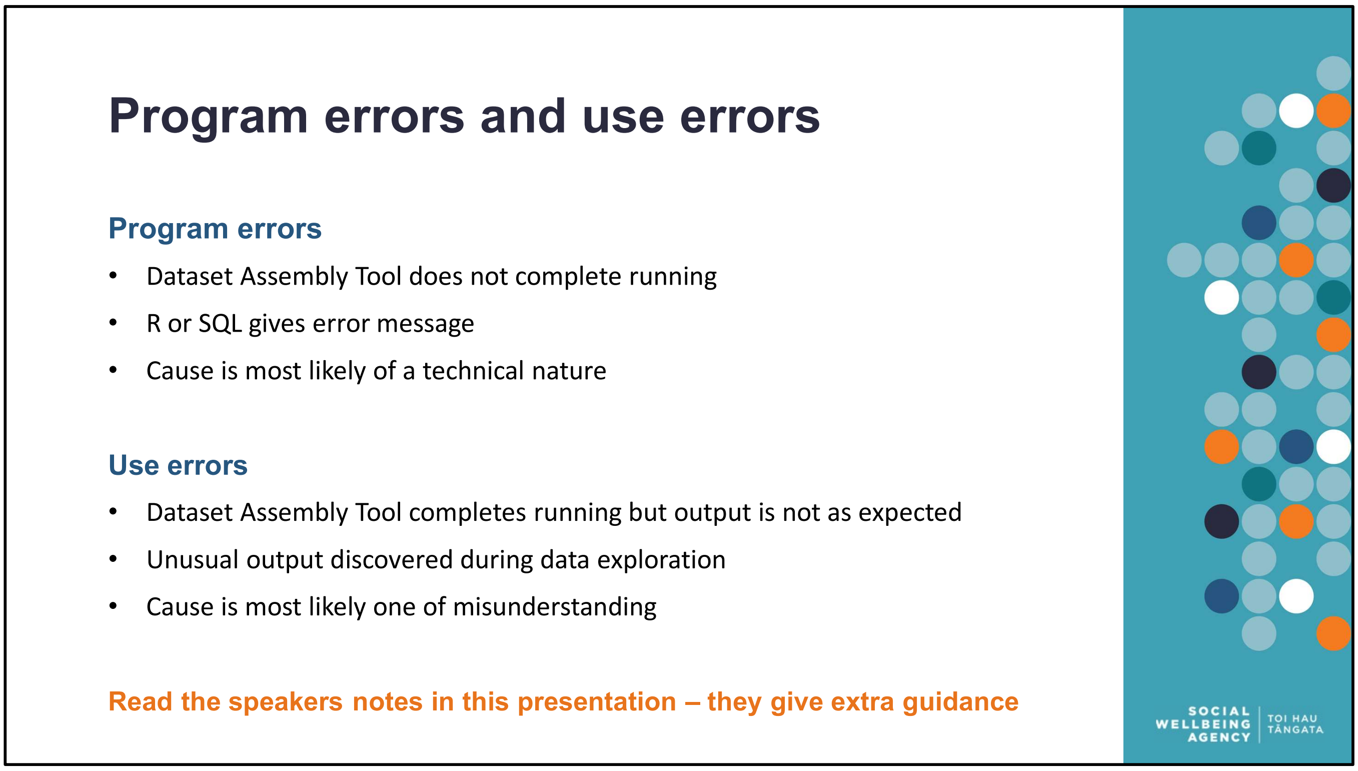


What to do if something goes wrong

Nothing is perfect, everything will break eventually. But most errors or problems should be fixable by the end user.
(Low dependence on an external expert, more self-service progress.)

Just because the presentation covers error handling in detail does NOT mean the tool is error prone. This section is detailed because so much effort has gone into ensuring the tool works smoothly for users.

Skip over this section in a short presentation.



Program errors and use errors

Program errors

- Dataset Assembly Tool does not complete running
- R or SQL gives error message
- Cause is most likely of a technical nature

Use errors

- Dataset Assembly Tool completes running but output is not as expected
- Unusual output discovered during data exploration
- Cause is most likely one of misunderstanding

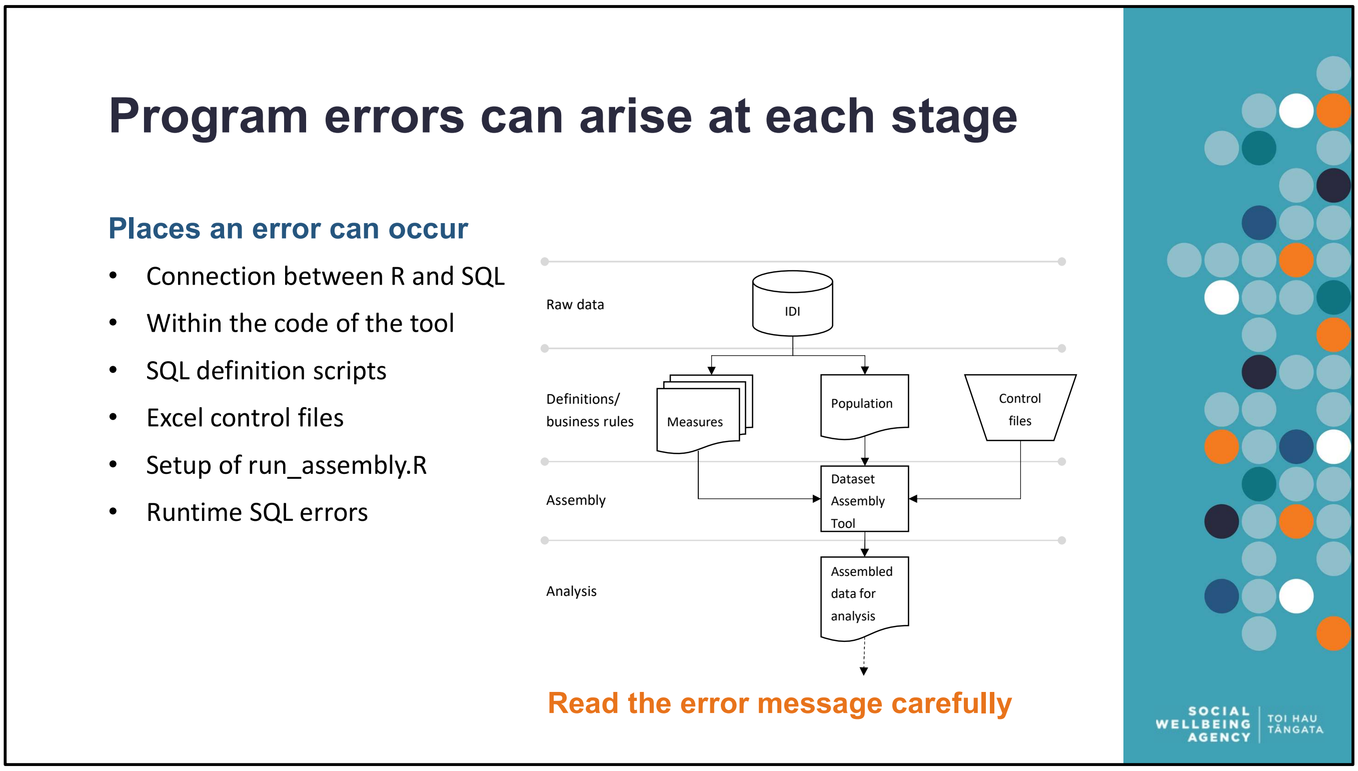
Read the speakers notes in this presentation – they give extra guidance

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Two broad reasons things can go wrong:
1. The computer attempts something it can not do → program errors
2. The computer does as instructed but does not give the desired output → use errors

Because the tool reads in definitions created by the user and control files created by the user there is a lot of scope for the user to give the tool invalid or faulty inputs. This section is designed to help users give correct inputs to the tool, and to correct their inputs when things do not work as expected.

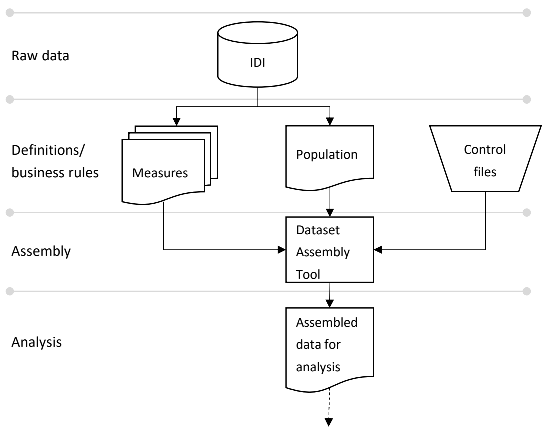
For some Use Errors, we have added checks to the tool that will give warnings or errors instead of producing unusual output. This blurs the difference between a Use Error and a Program Error.



Program errors can arise at each stage

Places an error can occur

- Connection between R and SQL
- Within the code of the tool
- SQL definition scripts
- Excel control files
- Setup of run_assembly.R
- Runtime SQL errors



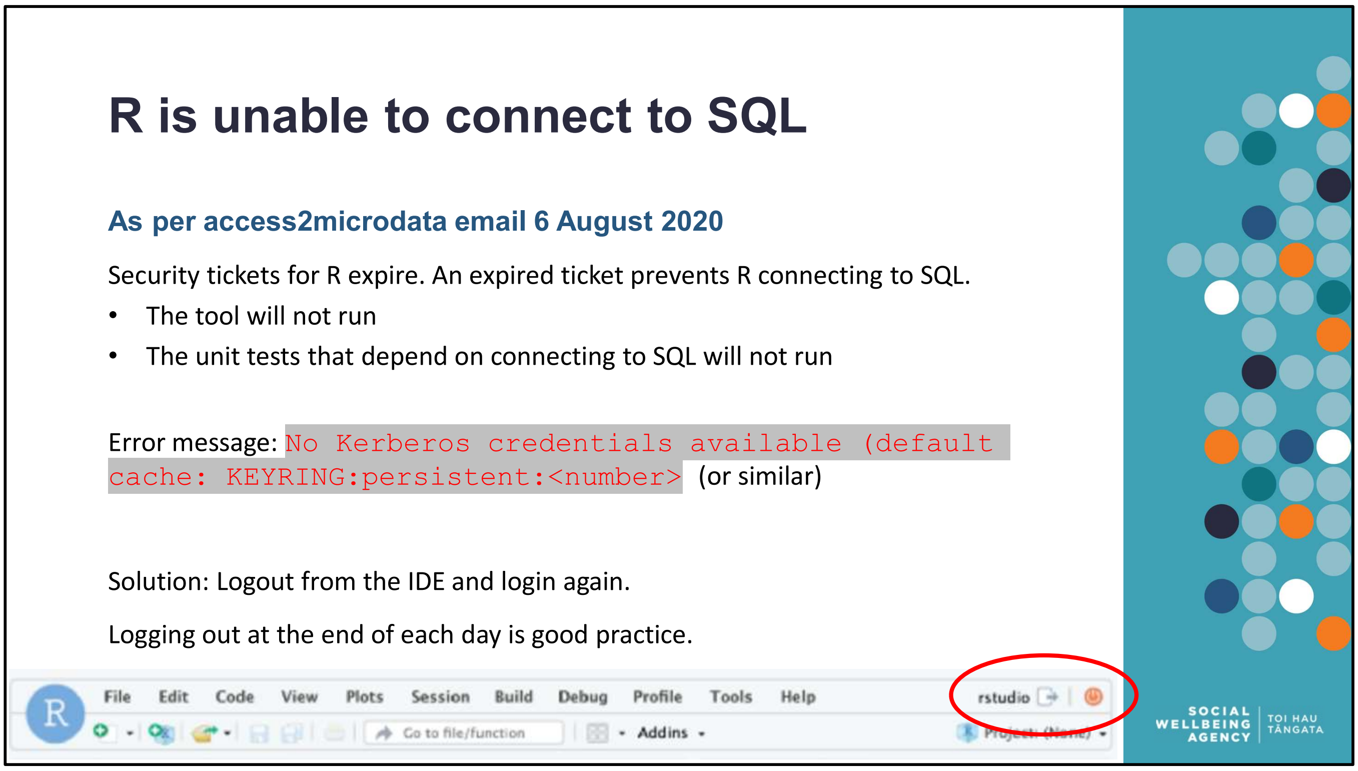
Read the error message carefully

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

There are a series of steps/stages that are involved in the assembling of data via the tool. Error are possible at each stage.

Program Errors at each of the different stages look different and have different solutions.
- Program Errors at all of the stages can be identified without R / programming knowledge.
- Program Errors at most of the stages can be fixed without R / programming knowledge.

Always read the error message, the tool has been designed to provide instructive error messages and user feedback.



R is unable to connect to SQL

As per access2microdata email 6 August 2020

Security tickets for R expire. An expired ticket prevents R connecting to SQL.

- The tool will not run
- The unit tests that depend on connecting to SQL will not run

Error message: `No Kerberos credentials available (default cache: KEYRING:persistent:<number>` (or similar)

Solution: Logout from the IDE and login again.

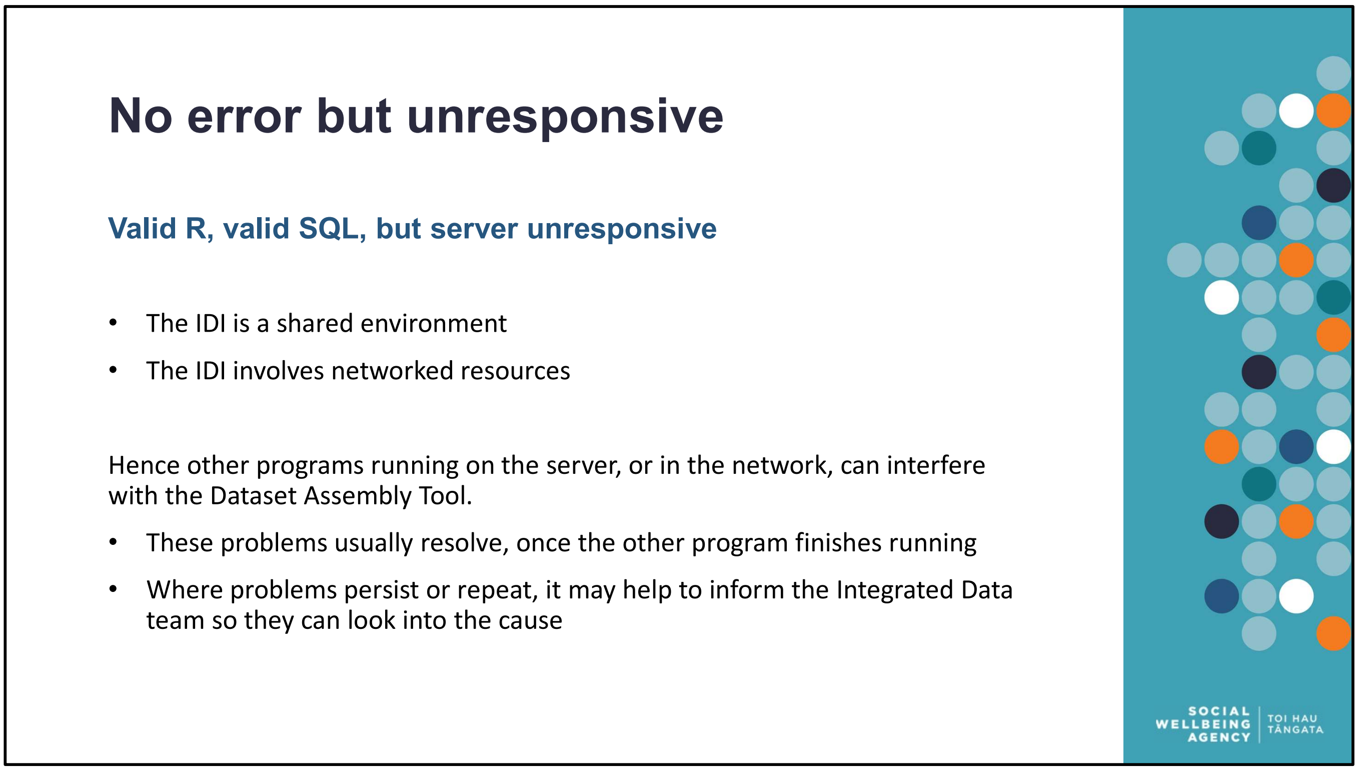
Logging out at the end of each day is good practice.

We most often encountered this error message when returning to R after leaving it logged in overnight.
This has been consistently resolved for us by (1) ending the current session (red 1/0 circle) and (2) logging out from RStudio (square & arrow symbol).
Both of these are good practice for finishing with & tidying up from an R session (releases R memory and restarts you with a clean environment).

Text from Access2microdata email 6 August 2020:

If you experience issues with ODBC:

1. If you get the error "No Kerberos credentials available (default cache: KEYRING:persistent:<number>)", then logout from the Rstudio Pro IDE and login again, as it is a security requirement that Kerberos tickets expire. Logging out at the end of the day is good practice, anyway.
2. Any errors referring to the ODBC driver version 11 not found, use: Driver = "ODBC Driver 17 for SQL Server".
3. Any other ODBC errors: check if you have a personal copy of an old odbc CRAN package. If so move it away. Then the system version of the odbc CRAN package (/usr/share/R/library/odbc) will be used, which has been proven to work.



No error but unresponsive

Valid R, valid SQL, but server unresponsive

- The IDI is a shared environment
- The IDI involves networked resources

Hence other programs running on the server, or in the network, can interfere with the Dataset Assembly Tool.

- These problems usually resolve, once the other program finishes running
- Where problems persist or repeat, it may help to inform the Integrated Data team so they can look into the cause

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Poor performance / non-responsiveness is unlikely to be an error, and more likely to be high traffic or large programs.
In August 2020, the most common problem of this nature was R becomes non-responsive because it is waiting for a response from SQL. If, while waiting for R, we went into SQL and requested a list of all Sandpit tables (a task that usually takes 1-2 seconds) we would instead receive an error message. This highlights that the problem is not with R or with the tool because SQL also errors.

Good checks to run:
1. Check you can use R without the tool
2. Check you can carry out equivalent work in SQL
3. Ask colleagues if they are experiencing the same or similar problems

When problems persist or repeat, you could inform the Integrated Data team in case they are unaware of the problem. When doing so we recommend providing details to assist their staff identify the cause (which may be you). This will not resolve the problem immediately, but it helps to improve the performance and robustness of the IDI environment – something that both Stats NZ and IDI researchers want.

Program Errors in the tool code

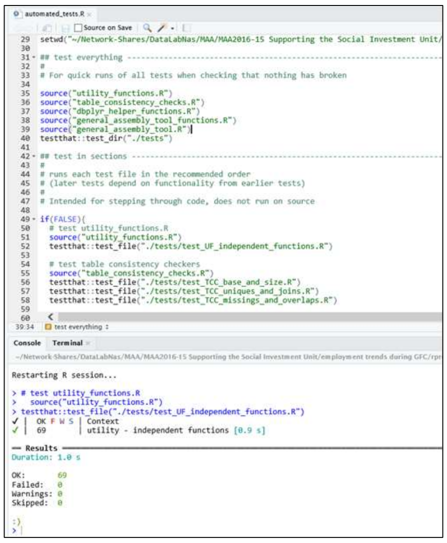
Automated tests

The assembly tool is covered by automated tests.

- Source the automated_tests.R file to run all tests
- If all the tests pass, the tool is working as designed

Tests can be run in sections

- This tests dependencies before dependents
- Run in sections to identify the point of failure



```
29  setwd("~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/
30
31  ## test everything ----------------------------------------------------------------
32  #
33  # For quick runs of all tests when checking that nothing has broken
34
35  source("utility_functions.R")
36  source("table_consistency_checks.R")
37  source("dbplyr_helper_functions.R")
38  source("general_assembly_tool_functions.R")
39  source("general_assembly_tool.R")
40  testthat::test_dir("./tests")
41
42  ## test in sections ----------------------------------------------------------------
43  #
44  # runs each test file in the recommended order
45  # (later tests depend on functionality from earlier tests)
46  #
47  # Intended for stepping through code, does not run on source
48
49  if(FALSE){
50    # test utility_functions.R
51    source("utility_functions.R")
52    testthat::test_file("./tests/test_UF_independent_functions.R")
53
54    # test table consistency checkers
55    source("table_consistency_checks.R")
56    testthat::test_file("./tests/test_TCC_base_and_size.R")
57    testthat::test_file("./tests/test_TCC_uniques_and_joins.R")
58    testthat::test_file("./tests/test_TCC_missings_and_overlaps.R")
59
60
```

```
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rpr

Restarting R session...

> # test utility_functions.R
>   source("utility_functions.R")
> testthat::test_file("./tests/test_UF_independent_functions.R")
✓ |  OK F W S | Context
✓ |  69       | utility - independent functions [0.9 s]

── Results ──────────────────────────────────────────
Duration: 1.0 s

OK:       69
Failed:   0
Warnings: 0
Skipped:  0

:)
>
```

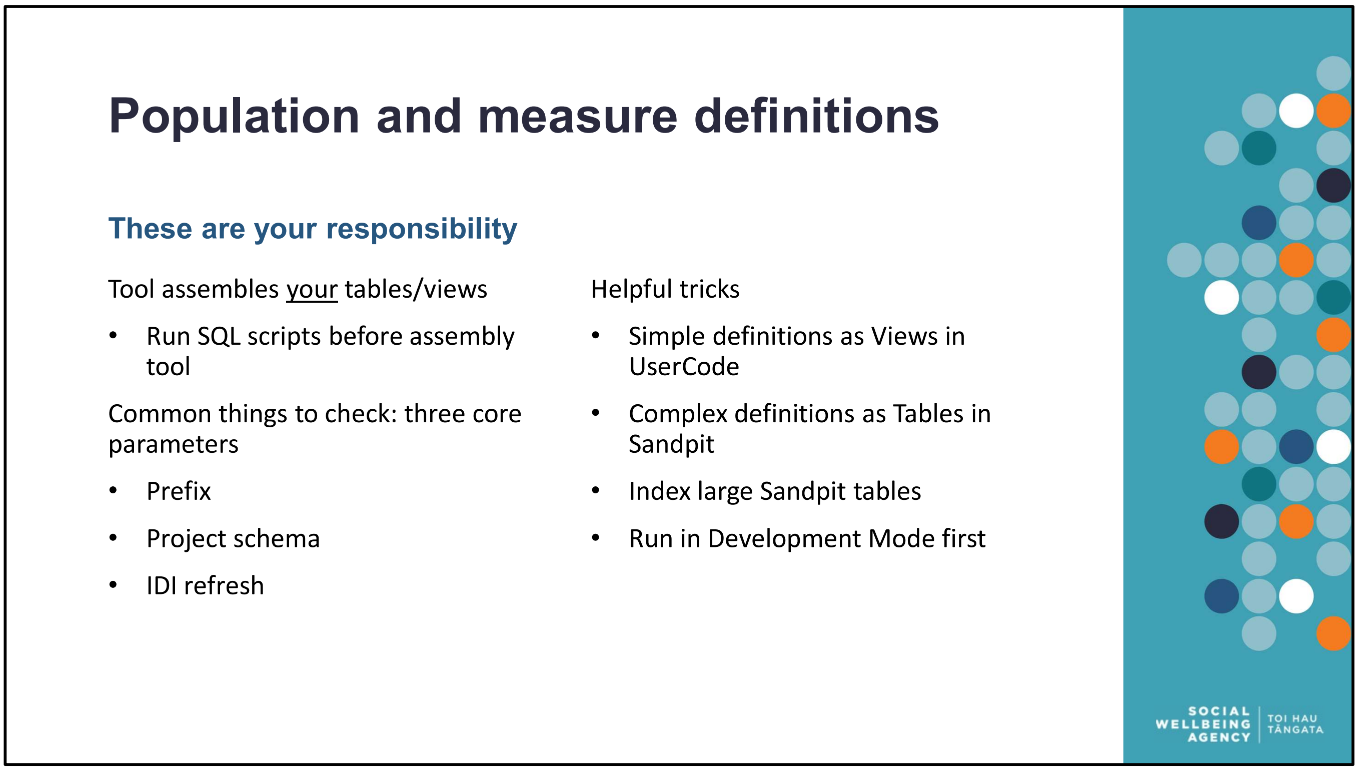
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Take a look at the image:

- The top third, lines 29-40, tests everything (but in no specific order). This code runs when the file is sourced.
- The middle third, lines 42-onwards, tests each section of the code in the order of dependency (later code will fail if earlier code does not pass). Run this code manually, line by line, is recommended if testing everything produces errors. This can be done by placing the cursor on the line you want to run and pressing Ctrl + Enter.

Note that line 49 is an if(FALSE) statement that prevents testing in sections running when testing everything. Do not run this line if you are trying to test in sections.

- The bottom third, the console, is an example of testing in sections. Here we have run lines 50, 51, and 52 to test the independent functions found in utility_functions.R



Population and measure definitions

These are your responsibility

Tool assembles your tables/views

- Run SQL scripts before assembly tool

Common things to check: three core parameters

- Prefix
- Project schema
- IDI refresh

Helpful tricks

- Simple definitions as Views in UserCode
- Complex definitions as Tables in Sandpit
- Index large Sandpit tables
- Run in Development Mode first

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

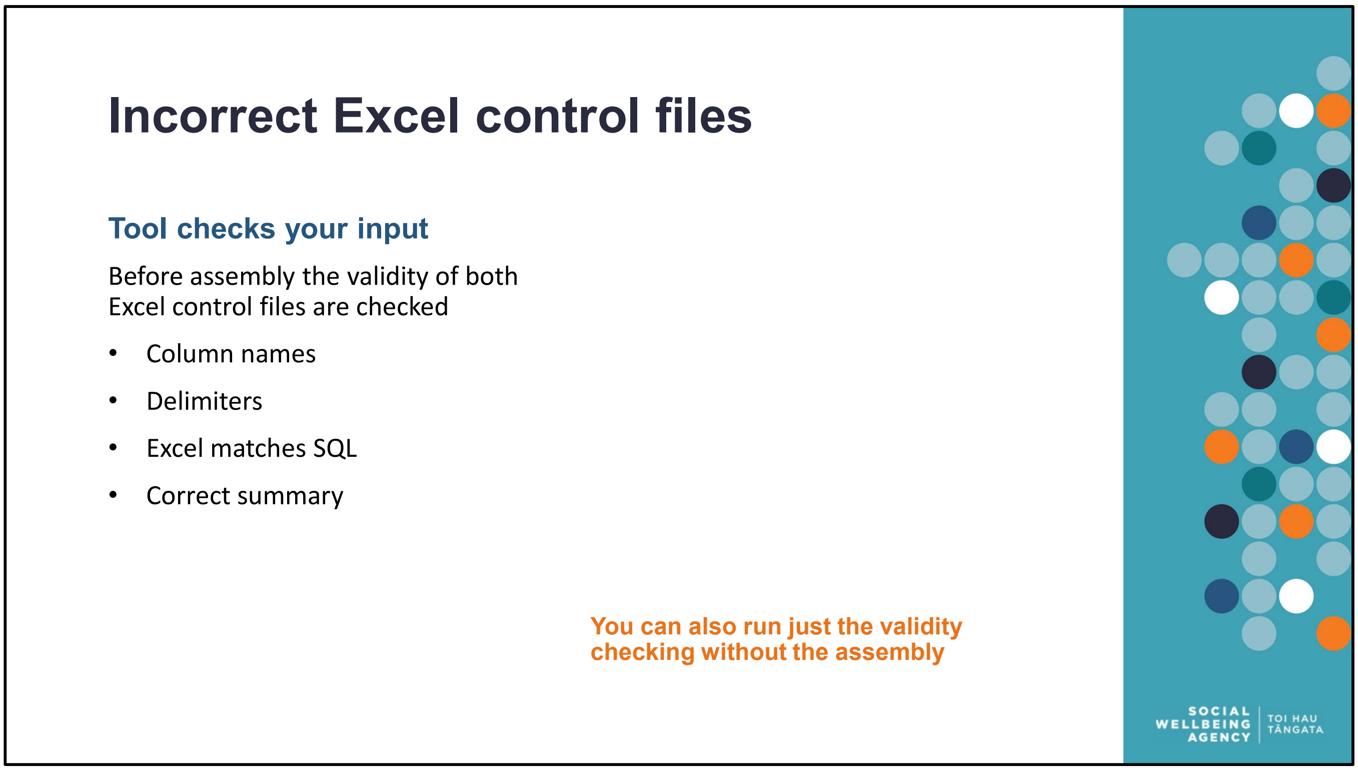
Population and measure definitions are defined in SQL. They are prepared and run before being included in the tool assembly.

- Make sure your SQL definition files run
- Make sure you run all your SQL definitions before running the assembly tool
- Check your definitions for
    - Correct prefix, schema, refresh
    - You have not accidentally restricted yourself to the “TOP 1000” rows

Just because a definition runs does not mean it will perform well.

- Simple definitions should be created as views – reduce memory space and make use of existing indexes
- Complex definitions should be created as tables with indexes

If the error is in your SQL definition (population or measure) script, then it is not an assembly tool related error.



Incorrect Excel control files

Tool checks your input

Before assembly the validity of both Excel control files are checked

- Column names
- Delimiters
- Excel matches SQL
- Correct summary

You can also run just the validity checking without the assembly

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

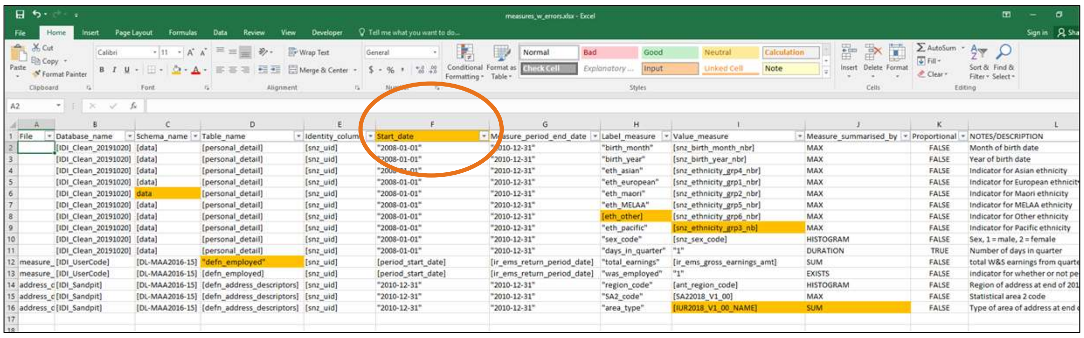
Before any dataset assembly happens the tool first checks that all your control file inputs are correct. Both control files are checked.
Four main checks are run:

1. The control files have all the required columns
2. All the entries in the required columns are correctly delimited
3. The SQL tables and columns specified in the control file exist in SQL
4. Summarising of variables is correct (e.g. you can't sum text)

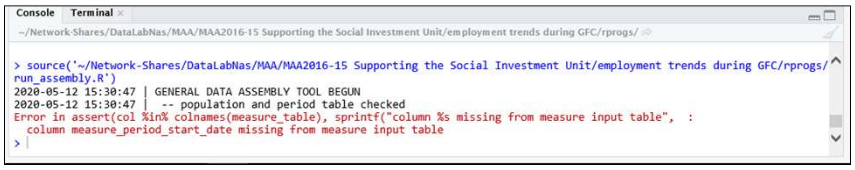
It will also give warnings if your summary input is inconsistent (e.g. there is no need to set Proportional = TRUE when counting).

You can run just the validity checking without the assembly. This helps with checking whether your Excel inputs are correct without having to wait for the assembly to run if they are.

Incorrect Excel (1) – column names



	A	B	C	D	E	F	G	H	I	J	K	L
1	File	Database_name	Schema_name	Table_name	Identity_colum	Start_date	Measure_period_end_date	Label_measure	Value_measure	Measure_summarised_by	Proportional	NOTES/DESCRIPTION
2		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_month"	[snz_birth_month_nbr]	MAX	FALSE	Month of birth date
3		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_year"	[snz_birth_year_nbr]	MAX	FALSE	Year of birth date
4		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_asian"	[snz_ethnicity_grp4_nbr]	MAX	FALSE	Indicator for Asian ethnicity
5		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_european"	[snz_ethnicity_grp1_nbr]	MAX	FALSE	Indicator for European ethnicity
6		[IDI_Clean_20191020]	data	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_maori"	[snz_ethnicity_grp2_nbr]	MAX	FALSE	Indicator for Maori ethnicity
7		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_MELAA"	[snz_ethnicity_grp5_nbr]	MAX	FALSE	Indicator for MELAA ethnicity
8		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	[eth_other]	[snz_ethnicity_grp6_nbr]	MAX	FALSE	Indicator for Other ethnicity
9		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_pacific"	[snz_ethnicity_grp3_nb]	MAX	FALSE	Indicator for Pacific ethnicity
10		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"sex_code"	[snz_sex_code]	HISTOGRAM	FALSE	Sex, 1 = male, 2 = female
11		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"days_in_quarter"	"1"	DURATION	TRUE	Number of days in quarter
12	measure_	[IDI_UserCode]	[DL-MAA2016-15]	"defn_employed"	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"total_earnings"	[ir_ems_gross_earnings_amt]	SUM	FALSE	total W&S earnings from quarte
13	measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"was_employed"	"1"	EXISTS	FALSE	indicator for whether or not pe
14	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"region_code"	[ant_region_code]	HISTOGRAM	FALSE	Region of address at end of 201
15	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"SA2_code"	[SA22018_V1_00]	MAX	FALSE	Statistical area 2 code
16	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"area_type"	[IUR2018_V1_00_NAME]	SUM	FALSE	Type of area of address at end c



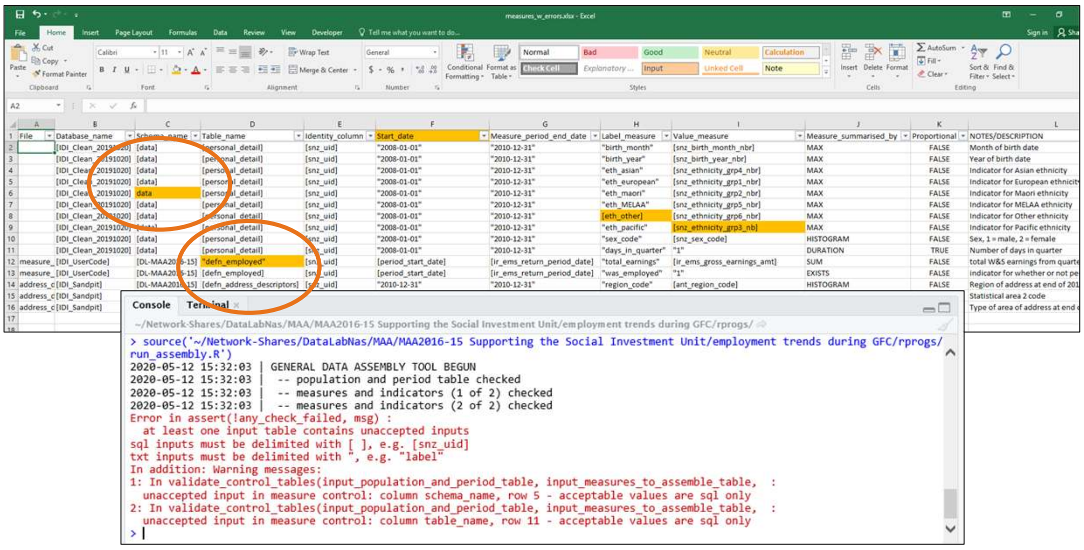
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/

```
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
run_assembly.R')
2020-05-12 15:30:47 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 15:30:47 |  -- population and period table checked
Error in assert(col %in% colnames(measure_table), sprintf("column %s missing from measure input table",  :
  column measure_period_start_date missing from measure input table
> 
```

Here is an example where the control file has an incorrect column name (but this would also happen if a column was missing).
The error message tells you that the column is missing (read the error messages).

Note that case (upper/lower) does not matter.
And that the File and Notes/description columns are optional.

Incorrect Excel (2) – delimiters



	File	Database_name	Schema_name	Table_name	Identity_column	Start_date	Measure_period_end_date	Label_measure	Value_measure	Measure_summarised_by	Proportional	NOTES/DESCRIPTION
2		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_month"	[snz_birth_month_nbr]	MAX	FALSE	Month of birth date
3		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_year"	[snz_birth_year_nbr]	MAX	FALSE	Year of birth date
4		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_asian"	[snz_ethnicity_grp4_nbr]	MAX	FALSE	Indicator for Asian ethnicity
5		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_european"	[snz_ethnicity_grp1_nbr]	MAX	FALSE	Indicator for European ethnicity
6		[IDI_Clean_20191020]	data	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_maori"	[snz_ethnicity_grp2_nbr]	MAX	FALSE	Indicator for Maori ethnicity
7		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_MELAA"	[snz_ethnicity_grp5_nbr]	MAX	FALSE	Indicator for MELAA ethnicity
8		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	[eth_other]	[snz_ethnicity_grp6_nbr]	MAX	FALSE	Indicator for Other ethnicity
9		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_pacific"	[snz_ethnicity_grp3_nb]	MAX	FALSE	Indicator for Pacific ethnicity
10		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"sex_code"	[snz_sex_code]	HISTOGRAM	FALSE	Sex, 1 = male, 2 = female
11		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"days_in_quarter"	"1"	DURATION	TRUE	Number of days in quarter
12	measure_	[IDI_UserCode]	[DL-MAA2016-15]	"defn_employed"	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"total_earnings"	[ir_ems_gross_earnings_amt]	SUM	FALSE	total W&S earnings from quarte
13	measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"was_employed"	"1"	EXISTS	FALSE	indicator for whether or not pe
14	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"region_code"	[ant_region_code]	HISTOGRAM	FALSE	Region of address at end of 201
15	address_c	[IDI_Sandpit]										Statistical area 2 code
16	address_c	[IDI_Sandpit]										Type of area of address at end o

```
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
run_assembly.R')
2020-05-12 15:32:03 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 15:32:03 |  -- population and period table checked
2020-05-12 15:32:03 |  -- measures and indicators (1 of 2) checked
2020-05-12 15:32:03 |  -- measures and indicators (2 of 2) checked
Error in assert(!any_check_failed, msg) :
  at least one input table contains unaccepted inputs
sql inputs must be delimited with [ ], e.g. [snz_uid]
txt inputs must be delimited with ", e.g. "label"
In addition: Warning messages:
1: In validate_control_tables(input_population_and_period_table, input_measures_to_assemble_table,  :
  unaccepted input in measure control: column schema_name, row 5 - acceptable values are sql only
2: In validate_control_tables(input_population_and_period_table, input_measures_to_assemble_table,  :
  unaccepted input in measure control: column table_name, row 11 - acceptable values are sql only
> |
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

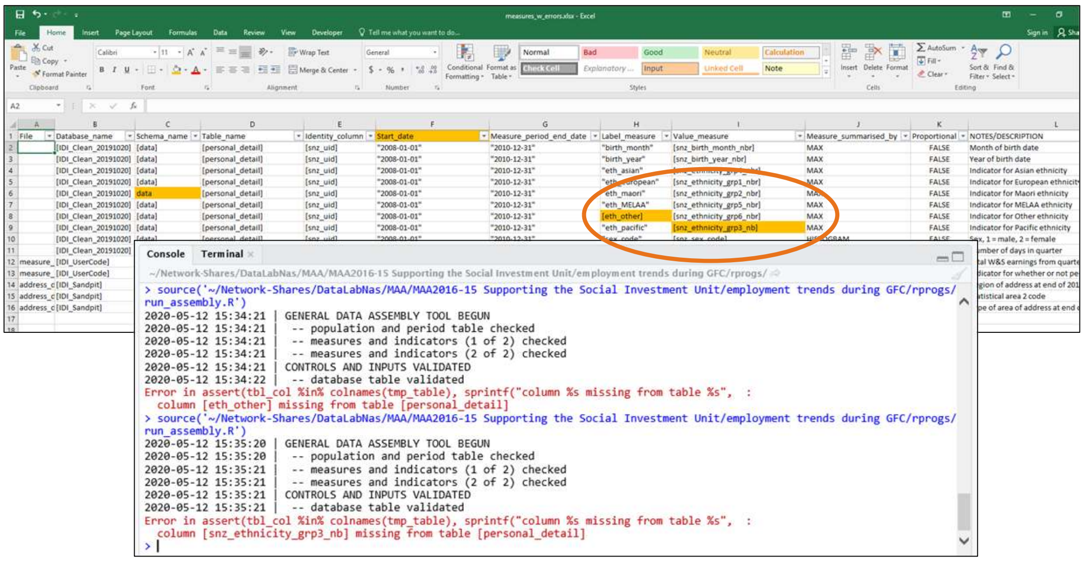
After fixing the previous error and rerunning, we get a new error message.

Here is an example where the delimiters are incorrect. In one case there are no delimiters, and in the other we have used " instead of [ ].
The error message tells you that delimiters are missing, how many, and where they are located.

Reminder:
- If the cell contents are SQL tables and columns then delimited [ ]
- If the cell contents are the value to use then delimit in "
- Measure_summarised_by and Proportional columns are not delimited

Note that double quotes is different from single quotes.
And straight double quotes are different from curly double quotes – if copy-and-paste from word check carefully that your quotes are straight not curly.

Incorrect Excel (3) – Excel matches SQL



	A	B	C	D	E	F	G	H	I	J	K	L
1	File	Database_name	Schema_name	Table_name	Identity_column	Start_date	Measure_period_end_date	Label_measure	Value_measure	Measure_summarised_by	Proportional	NOTES/DESCRIPTION
2		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_month"	[snz_birth_month_nbr]	MAX	FALSE	Month of birth date
3		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_year"	[snz_birth_year_nbr]	MAX	FALSE	Year of birth date
4		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_asian"	[illegible]	MAX	FALSE	Indicator for Asian ethnicity
5		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_european"	[snz_ethnicity_grp1_nbr]	MAX	FALSE	Indicator for European ethnicit
6		[IDI_Clean_20191020]	data	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_maori"	[snz_ethnicity_grp2_nbr]	MAX	FALSE	Indicator for Maori ethnicity
7		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_MELAA"	[snz_ethnicity_grp5_nbr]	MAX	FALSE	Indicator for MELAA ethnicity
8		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	[eth_other]	[snz_ethnicity_grp6_nbr]	MAX	FALSE	Indicator for Other ethnicity
9		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_pacific"	[snz_ethnicity_grp3_nb]	MAX	FALSE	Indicator for Pacific ethnicity
10		[IDI_Clean_20191020]										Sex, 1 = male, 2 = female
11		[IDI_Clean_20191020]										umber of days in quarter
12	measure_	[IDI_UserCode]										tal W&S earnings from quarte
13	measure_	[IDI_UserCode]										dicator for whether or not pe
14	address_c	[IDI_Sandpit]										gion of address at end of 201
15	address_c	[IDI_Sandpit]										atistical area 2 code
16	address_c	[IDI_Sandpit]										pe of area of address at end
17												

```
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
run_assembly.R')
2020-05-12 15:34:21 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 15:34:21 |  -- population and period table checked
2020-05-12 15:34:21 |  -- measures and indicators (1 of 2) checked
2020-05-12 15:34:21 |  -- measures and indicators (2 of 2) checked
2020-05-12 15:34:21 | CONTROLS AND INPUTS VALIDATED
2020-05-12 15:34:22 |  -- database table validated
Error in assert(tbl_col %in% colnames(tmp_table), sprintf("column %s missing from table %s",  :
  column [eth_other] missing from table [personal_detail]
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
run_assembly.R')
2020-05-12 15:35:20 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 15:35:20 |  -- population and period table checked
2020-05-12 15:35:21 |  -- measures and indicators (1 of 2) checked
2020-05-12 15:35:21 |  -- measures and indicators (2 of 2) checked
2020-05-12 15:35:21 | CONTROLS AND INPUTS VALIDATED
2020-05-12 15:35:21 |  -- database table validated
Error in assert(tbl_col %in% colnames(tmp_table), sprintf("column %s missing from table %s",  :
  column [snz_ethnicity_grp3_nb] missing from table [personal_detail]
> |
```

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

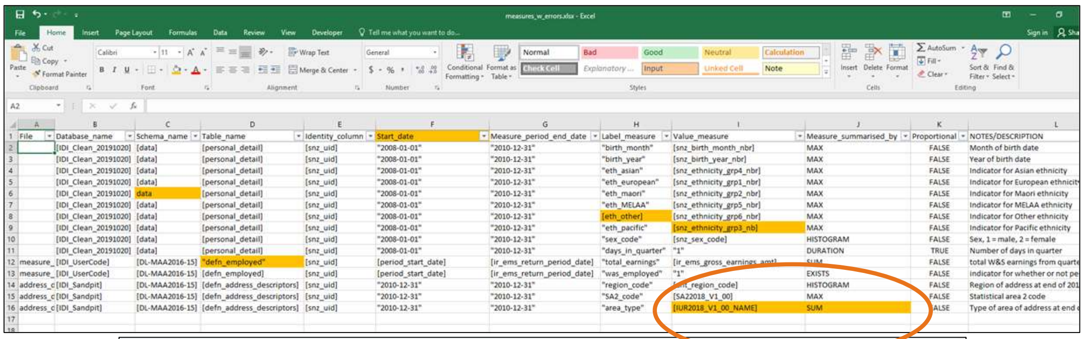
After fixing the previous error and rerunning, we get a new error message.

Here is an example where the tool can not find the columns specified in SQL.
- For [eth_other] this is because there is no column [eth_other] this should be a label “eth_other” and is incorrectly delimited
- For [snz_ethnicity_grp3_nb] this is a typo, the column name has been incorrectly entered (it should end _nbr] not _nb])

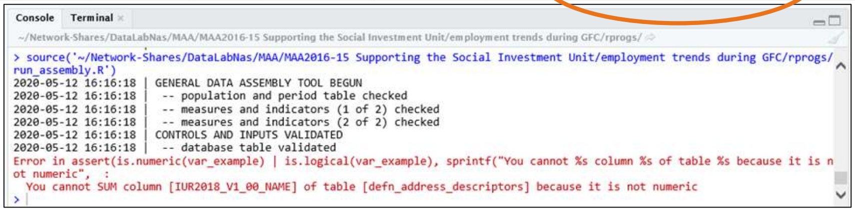
The error message gives you the value of the cell that can not be found.

If you program SQL with square brackets (as SQL defaults to when you ask to view the top 1000 rows) then the best way I have found of populating these cells is:
- double-click on a variable in SQL
- Copy
- and paste into Excel

Incorrect Excel (4) – correct summary



	File	Database_name	Schema_name	Table_name	Identity_column	Start_date	Measure_period_end_date	Label_measure	Value_measure	Measure_summarised_by	Proportional	NOTES/DESCRIPTION
2		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_month"	[snz_birth_month_nbr]	MAX	FALSE	Month of birth date
3		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"birth_year"	[snz_birth_year_nbr]	MAX	FALSE	Year of birth date
4		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_asian"	[snz_ethnicity_grp4_nbr]	MAX	FALSE	Indicator for Asian ethnicity
5		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_european"	[snz_ethnicity_grp1_nbr]	MAX	FALSE	Indicator for European ethnicity
6		[IDI_Clean_20191020]	data	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_maori"	[snz_ethnicity_grp2_nbr]	MAX	FALSE	Indicator for Maori ethnicity
7		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_MELAA"	[snz_ethnicity_grp5_nbr]	MAX	FALSE	Indicator for MELAA ethnicity
8		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	[eth_other]	[snz_ethnicity_grp6_nbr]	MAX	FALSE	Indicator for Other ethnicity
9		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"eth_pacific"	[snz_ethnicity_grp3_nb]	MAX	FALSE	Indicator for Pacific ethnicity
10		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"sex_code"	[snz_sex_code]	HISTOGRAM	FALSE	Sex, 1 = male, 2 = female
11		[IDI_Clean_20191020]	[data]	[personal_detail]	[snz_uid]	"2008-01-01"	"2010-12-31"	"days_in_quarter"	"1"	DURATION	TRUE	Number of days in quarter
12	measure_	[IDI_UserCode]	[DL-MAA2016-15]	"defn_employed"	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"total_earnings"	[ir_ems_gross_earnings_amt]	SUM	FALSE	total W&S earnings from quarte
13	measure_	[IDI_UserCode]	[DL-MAA2016-15]	[defn_employed]	[snz_uid]	[period_start_date]	[ir_ems_return_period_date]	"was_employed"	"1"	EXISTS	FALSE	indicator for whether or not pe
14	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"region_code"	[illegible]t_region_code]	HISTOGRAM	FALSE	Region of address at end of 201
15	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"SA2_code"	[SA22018_V1_00]	MAX	FALSE	Statistical area 2 code
16	address_c	[IDI_Sandpit]	[DL-MAA2016-15]	[defn_address_descriptors]	[snz_uid]	"2010-12-31"	"2010-12-31"	"area_type"	[IUR2018_V1_00_NAME]	SUM	ALSE	Type of area of address at end c



```
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
> source('~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
run_assembly.R')
2020-05-12 16:16:18 | GENERAL DATA ASSEMBLY TOOL BEGUN
2020-05-12 16:16:18 |  -- population and period table checked
2020-05-12 16:16:18 |  -- measures and indicators (1 of 2) checked
2020-05-12 16:16:18 |  -- measures and indicators (2 of 2) checked
2020-05-12 16:16:18 | CONTROLS AND INPUTS VALIDATED
2020-05-12 16:16:18 |  -- database table validated
Error in assert(is.numeric(var_example) | is.logical(var_example), sprintf("You cannot %s column %s of table %s because it is n
ot numeric",  :
  You cannot SUM column [IUR2018_V1_00_NAME] of table [defn_address_descriptors] because it is not numeric
>
```

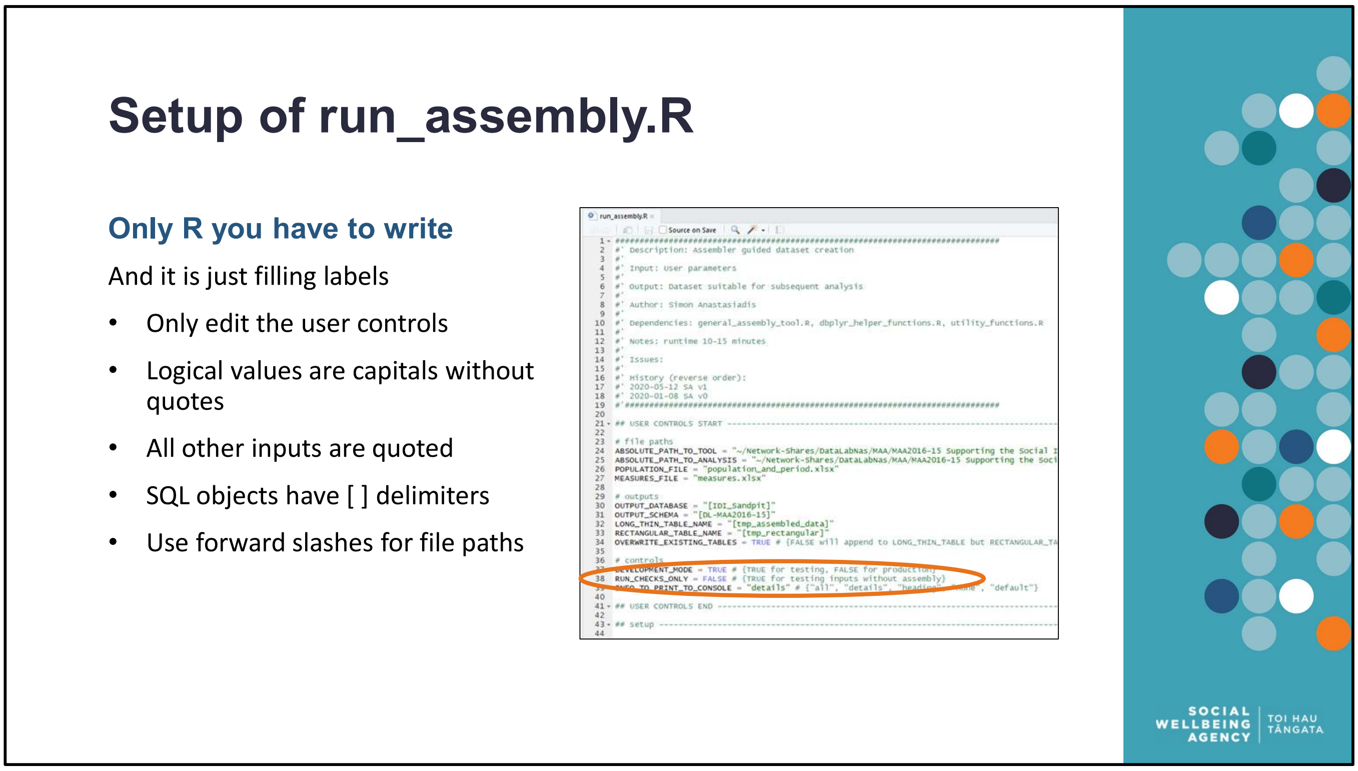
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

After fixing the previous error and rerunning, we get a new error message.

Here is an example where the contents of the column containing the value_measure are not compatible with the measure_summarised_by that has been chosen.
The error message tells you what column, what command, and why.

Many codes in the IDI are stored as text rather than as numeric. This is the most common cause I have encountered. Two solutions:

- Cast the variable to numeric in SQL: CAST(COL AS TYPE) AS new_name
- Use the histogram summary instead

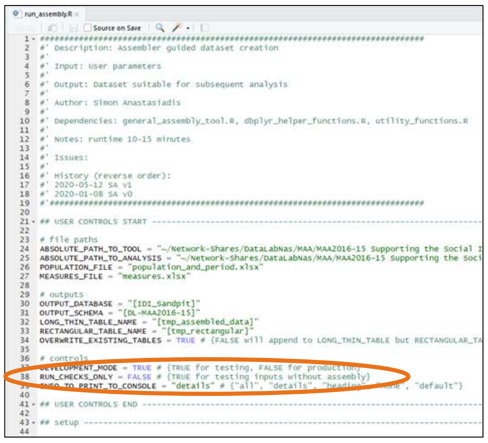


Setup of run_assembly.R

Only R you have to write

And it is just filling labels

- Only edit the user controls
- Logical values are capitals without quotes
- All other inputs are quoted
- SQL objects have [ ] delimiters
- Use forward slashes for file paths

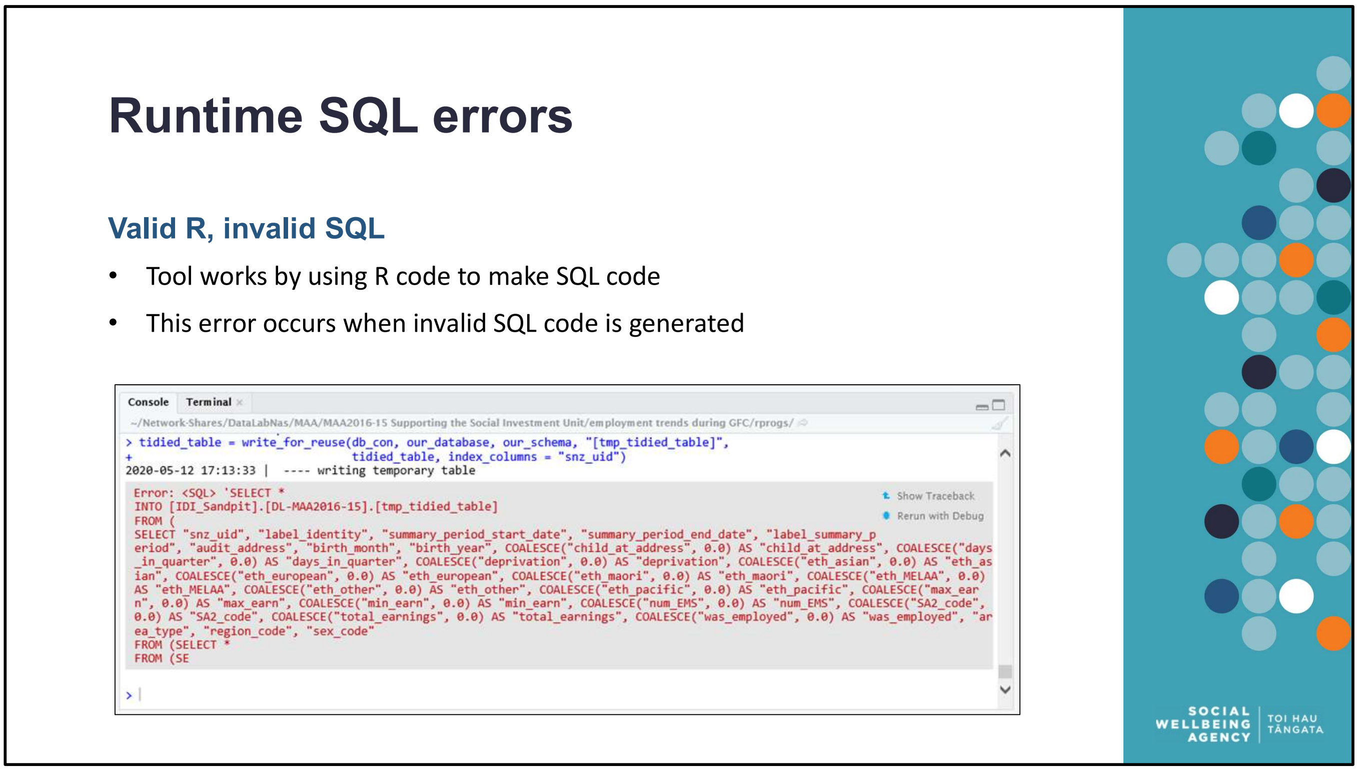


```
 1 ################################################################################
 2 #' Description: Assembler guided dataset creation
 3 #'
 4 #' Input: User parameters
 5 #'
 6 #' Output: Dataset suitable for subsequent analysis
 7 #'
 8 #' Author: Simon Anastasiadis
 9 #'
10 #' Dependencies: general_assembly_tool.R, dbplyr_helper_functions.R, utility_functions.R
11 #'
12 #' Notes: runtime 10-15 minutes
13 #'
14 #' Issues:
15 #'
16 #' History (reverse order):
17 #' 2020-05-12 SA v1
18 #' 2020-01-08 SA v0
19 #'##############################################################################
20 
21 ## USER CONTROLS START ----------------------------------------------------------
22 
23 # file paths
24 ABSOLUTE_PATH_TO_TOOL = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social I
25 ABSOLUTE_PATH_TO_ANALYSIS = "~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Soci
26 POPULATION_FILE = "population_and_period.xlsx"
27 MEASURES_FILE = "measures.xlsx"
28 
29 # outputs
30 OUTPUT_DATABASE = "[IDI_Sandpit]"
31 OUTPUT_SCHEMA = "[DL-MAA2016-15]"
32 LONG_THIN_TABLE_NAME = "[tmp_assembled_data]"
33 RECTANGULAR_TABLE_NAME = "[tmp_rectangular]"
34 OVERWRITE_EXISTING_TABLES = TRUE # {FALSE will append to LONG_THIN_TABLE but RECTANGULAR_TA
35 
36 # controls
37 DEVELOPMENT_MODE = TRUE # {TRUE for testing, FALSE for production}
38 RUN_CHECKS_ONLY = FALSE # {TRUE for testing inputs without assembly}
39 INFO_TO_PRINT_TO_CONSOLE = "details" # {"all", "details", "heading", "none", "default"}
40 
41 ## USER CONTROLS END ------------------------------------------------------------
42 
43 ## setup ------------------------------------------------------------------------
44 
```

This is the only piece of R code you have to edit, and all you are doing is entering your inputs.

- Only edit within the section labelled user controls (lines 21-40)
- Logical values are in capitals, no quotes
- Everything else is quoted
- SQL objects (database, schema, table names) have [ ] delimiters without their quotes (e.g. “[output_database]”)

Careful with slashes in file paths. Windows uses backward \ slashes, but R uses forward / slashes. This will cause errors if copy-and-paste from windows without editing.



Runtime SQL errors

Valid R, invalid SQL

- Tool works by using R code to make SQL code
- This error occurs when invalid SQL code is generated

Console
Terminal
~/Network-Shares/DataLabNas/MAA/MAA2016-15 Supporting the Social Investment Unit/employment trends during GFC/rprogs/
Show Traceback
Rerun with Debug

```
> tidied_table = write_for_reuse(db_con, our_database, our_schema, "[tmp_tidied_table]",
+                               tidied_table, index_columns = "snz_uid")
2020-05-12 17:13:33 |  ---- writing temporary table
 Error: <SQL> 'SELECT *
 INTO [IDI_Sandpit].[DL-MAA2016-15].[tmp_tidied_table]
 FROM (
 SELECT "snz_uid", "label_identity", "summary_period_start_date", "summary_period_end_date", "label_summary_p
 eriod", "audit_address", "birth_month", "birth_year", COALESCE("child_at_address", 0.0) AS "child_at_address", COALESCE("days
 _in_quarter", 0.0) AS "days_in_quarter", COALESCE("deprivation", 0.0) AS "deprivation", COALESCE("eth_asian", 0.0) AS "eth_as
 ian", COALESCE("eth_european", 0.0) AS "eth_european", COALESCE("eth_maori", 0.0) AS "eth_maori", COALESCE("eth_MELAA", 0.0)
 AS "eth_MELAA", COALESCE("eth_other", 0.0) AS "eth_other", COALESCE("eth_pacific", 0.0) AS "eth_pacific", COALESCE("max_ear
 n", 0.0) AS "max_earn", COALESCE("min_earn", 0.0) AS "min_earn", COALESCE("num_EMS", 0.0) AS "num_EMS", COALESCE("SA2_code",
 0.0) AS "SA2_code", COALESCE("total_earnings", 0.0) AS "total_earnings", COALESCE("was_employed", 0.0) AS "was_employed", "ar
 ea_type", "region_code", "sex_code"
 FROM (SELECT *
 FROM (SE
> |
```

Sometimes the combination of the input definitions (population and measure) and the SQL code generated by the assembly tool produce errors.

Note how the message starts with "Error: <SQL>" this is a clear sign of a Runtime SQL error. Also observe that the error happened while writing a table. Because writing a table is an SQL activity, this also tells us the error occurred in SQL.

Possibilities include:

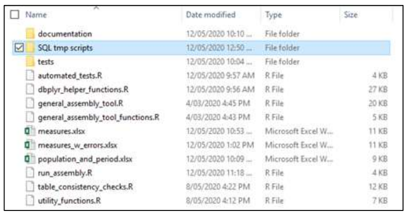
1. Attempting to SUM a column that is not numeric → The SQL definition and control file both seem correct, but when combining them (text column from SQL definition, SUM instruction from control file) you get an error (the tool now checks for this).
2. Too many variables → SQL has a maximum of 1024 columns, so if you create lots of measures (or use HISTOGRAM a lot) you can cause this error. You will need to reduce the number of measures or find a better way to express you current measures.
3. Numeric overflow → SQL has a maximum sized integer number it can consider, so if you create a very large value (such as total income ever recorded for all people in the South Island) you can cause this error. You will need to scale down your value (calculate in $millions) or find another way to measure it.

“SQL tmp scripts” folder

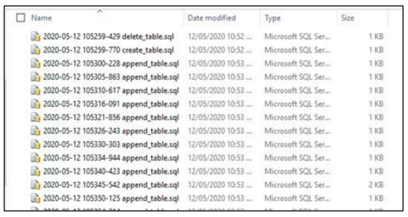
Review in SQL

R tells you where the error occurs, but not why

- Open “SQL tmp scripts” folder
- Run the latest code file in SQL Server Management Studio
- Determine from the error message where the problem is



Name	Date modified	Type	Size
documentation	12/05/2020 10:10 ...	File folder	
SQL tmp scripts	12/05/2020 12:50 ...	File folder	
tests	12/05/2020 10:04 ...	File folder	
automated_tests.R	12/05/2020 9:57 AM	R File	4 KB
dbplyr_helper_functions.R	12/05/2020 9:56 AM	R File	27 KB
general_assembly_tool.R	4/03/2020 4:45 PM	R File	20 KB
general_assembly_tool_functions.R	4/03/2020 4:43 PM	R File	5 KB
measures.xlsx	12/05/2020 10:53 ...	Microsoft Excel W...	11 KB
measures_w_errors.xlsx	12/05/2020 1:02 PM	Microsoft Excel W...	11 KB
population_and_period.xlsx	12/05/2020 10:09 ...	Microsoft Excel W...	9 KB
run_assembly.R	12/05/2020 11:18 ...	R File	4 KB
table_consistency_checks.R	8/05/2020 4:22 PM	R File	12 KB
utility_functions.R	8/05/2020 4:12 PM	R File	7 KB



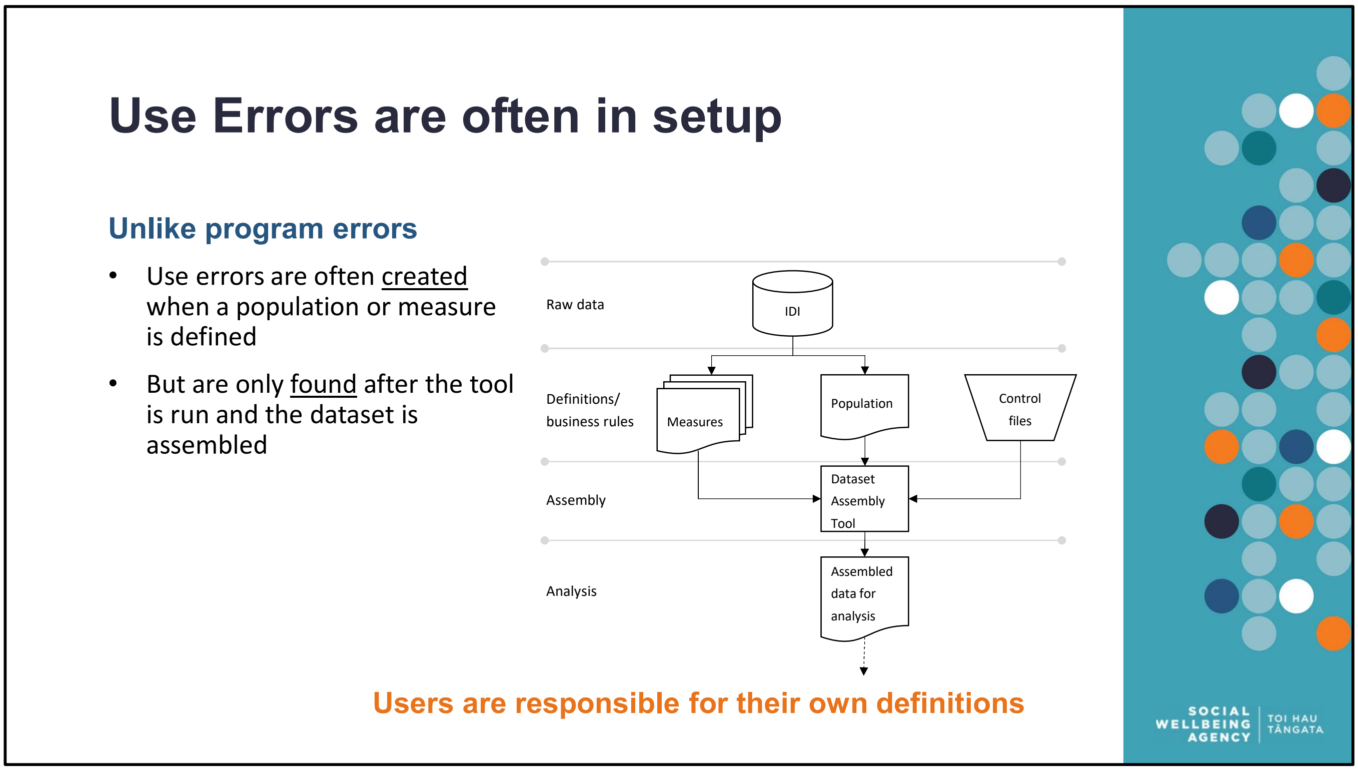
Name	Date modified	Type	Size
2020-05-12 105259-429 delete_table.sql	12/05/2020 10:52 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105259-770 create_table.sql	12/05/2020 10:52 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105300-228 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105305-863 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105310-617 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105316-091 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105321-856 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105326-243 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105330-303 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105334-944 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105340-423 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB
2020-05-12 105345-542 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	2 KB
2020-05-12 105350-125 append_table.sql	12/05/2020 10:53 ...	Microsoft SQL Ser...	1 KB

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

When this happens find the “SQL tmp scripts” folder. Every (non-trivial) SQL command executed by the tool gets saved in this folder. Sort the files and open the latest one in SQL Server Management Studio. This should be the code that caused the error. Test whether the code can be run directly from SQL.

- If it runs fine, then there may not be an error – just a random hiccup – try running the assembly tool again
- If it takes a very long time to run, then it probably takes too long to run for R and the connection is timing out
- If it creates errors, then further inspection is necessary to determine which part of the query is causing the error

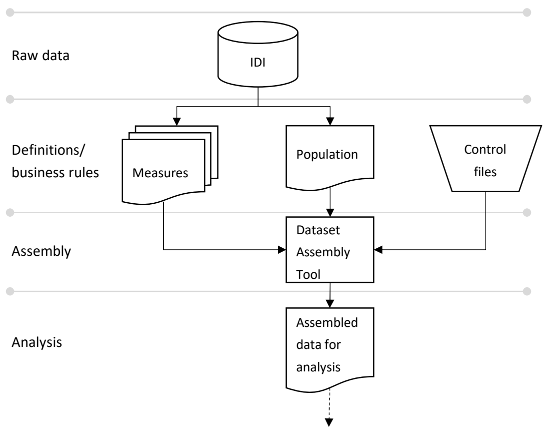
You can delete the “SQL tmp scripts” folder without risk – it will recreate itself. We recommend that you delete the folder regularly as it is only there to support debugging / logging and can accumulate thousands of little files which have little value.



Use Errors are often in setup

Unlike program errors

- Use errors are often created when a population or measure is defined
- But are only found after the tool is run and the dataset is assembled



Users are responsible for their own definitions

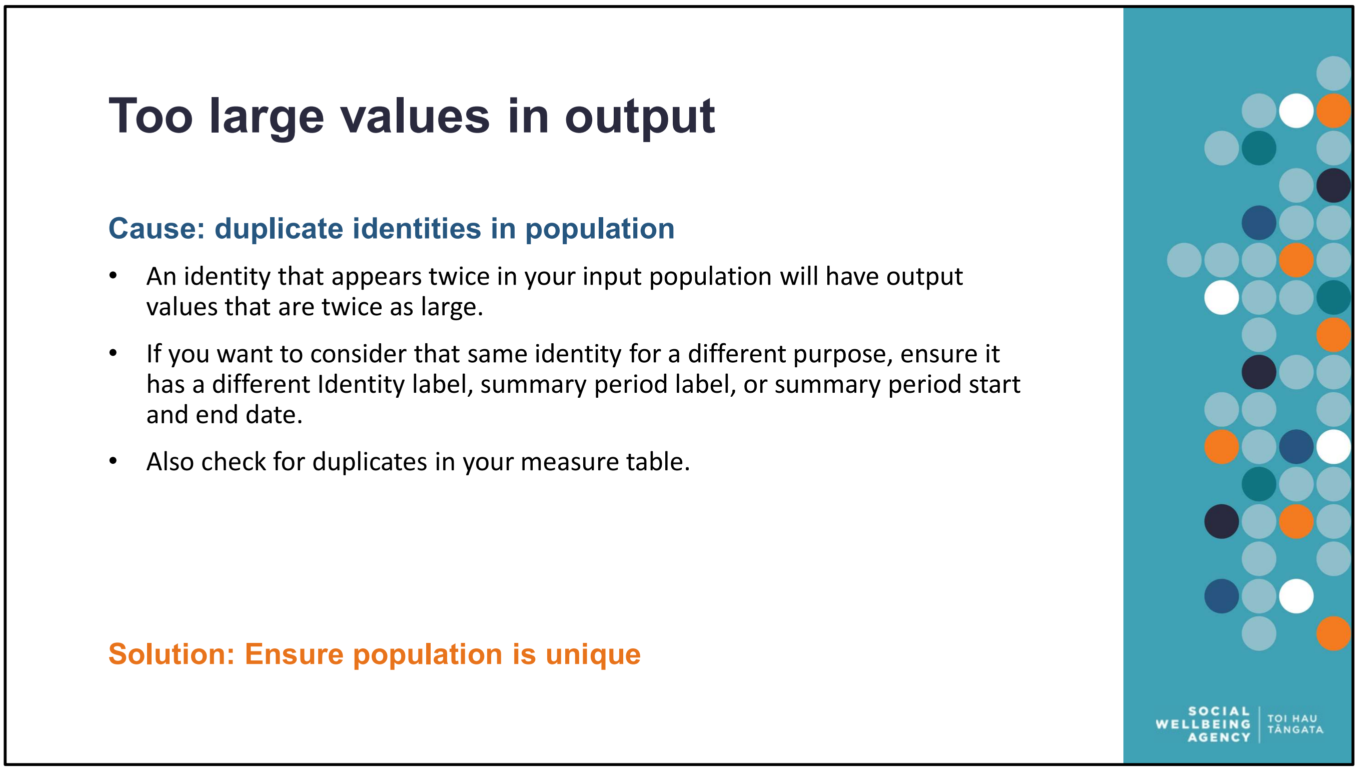
Once output is assembled, it is important to assess the prepared table before it is used in analysis. This often happens as part of tidying up the dataset for analysis (handling missing values, reclassifying variables, removing identities with errors, etc.).

It is not always possible to spot Use Errors from the reduced output produced when the tool is run in DEVELOPMENT MODE. You will often have to fully assemble the dataset before you can check for use error.

Where the assembled dataset contains output that is different from expectations, then further investigation will likely be required.

- Auditing (manually reconstructing) the values for a single identity an effective way to identify user error
- When high standards of quality are critical, you may want to audit the assembled dataset even if there is no sign of use error

In the following slides, we discuss several types of use error that we are already aware of. More examples may be added later as the number of people using the tool increases.



Too large values in output

Cause: duplicate identities in population

- An identity that appears twice in your input population will have output values that are twice as large.
- If you want to consider that same identity for a different purpose, ensure it has a different Identity label, summary period label, or summary period start and end date.
- Also check for duplicates in your measure table.

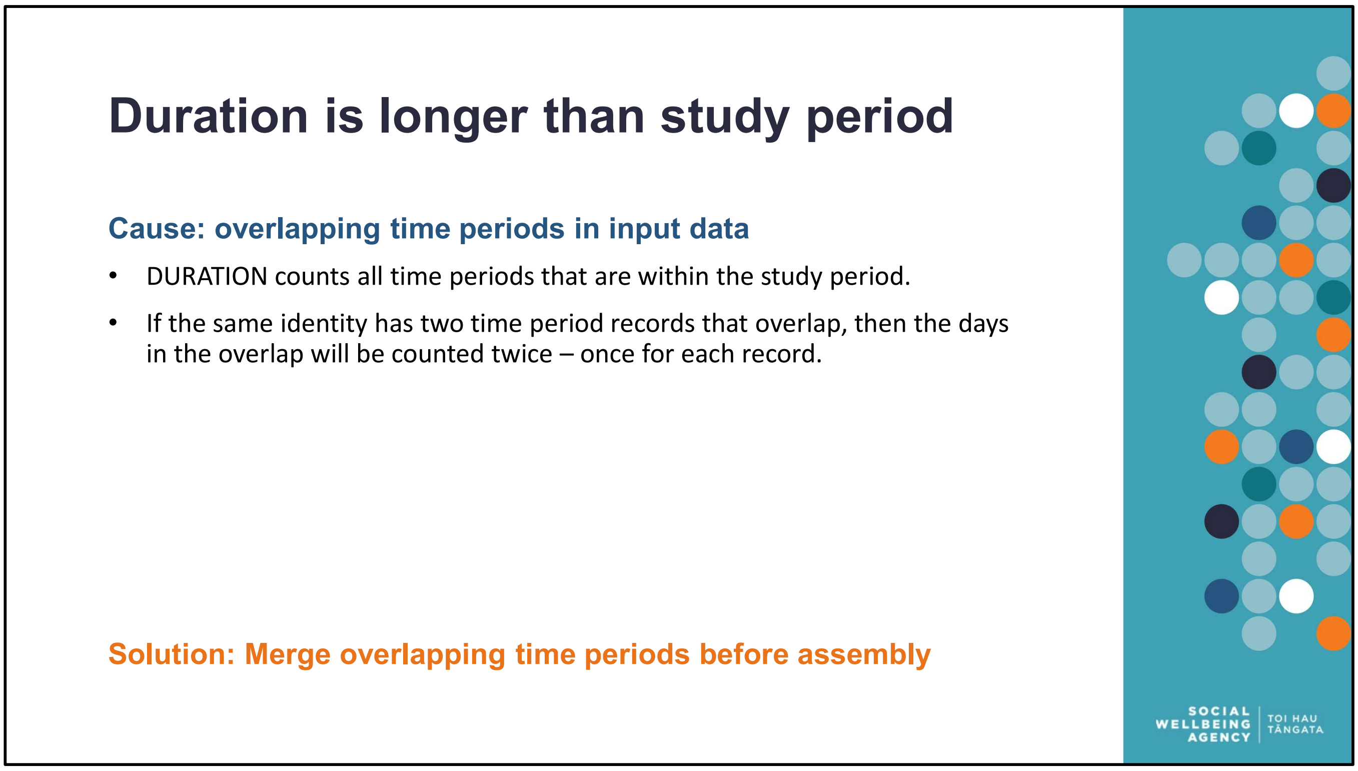
Solution: Ensure population is unique

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Because the population table and the measure tables are joined using “INNER JOIN” duplicates in either the population or measure table will get picked up more than once in your output.

Example: Research considers mothers who have a baby born in 2017 or 2018. Initial input population is all mothers in MOH maternity records with year given birth 2017 or 2018. What about mothers who gave birth two (or more) times?

1. Do nothing → Duplicate output for these mothers
2. De-duplicate the population → One output row for each mother, but mothers who gave birth twice could have twice as many events as mothers who only gave birth once
3. Set the summary period start and end date based on the date of the birth → Mothers who gave birth twice appear twice in output dataset, once for each set of dates



Duration is longer than study period

Cause: overlapping time periods in input data

- DURATION counts all time periods that are within the study period.
- If the same identity has two time period records that overlap, then the days in the overlap will be counted twice – once for each record.

Solution: Merge overlapping time periods before assembly

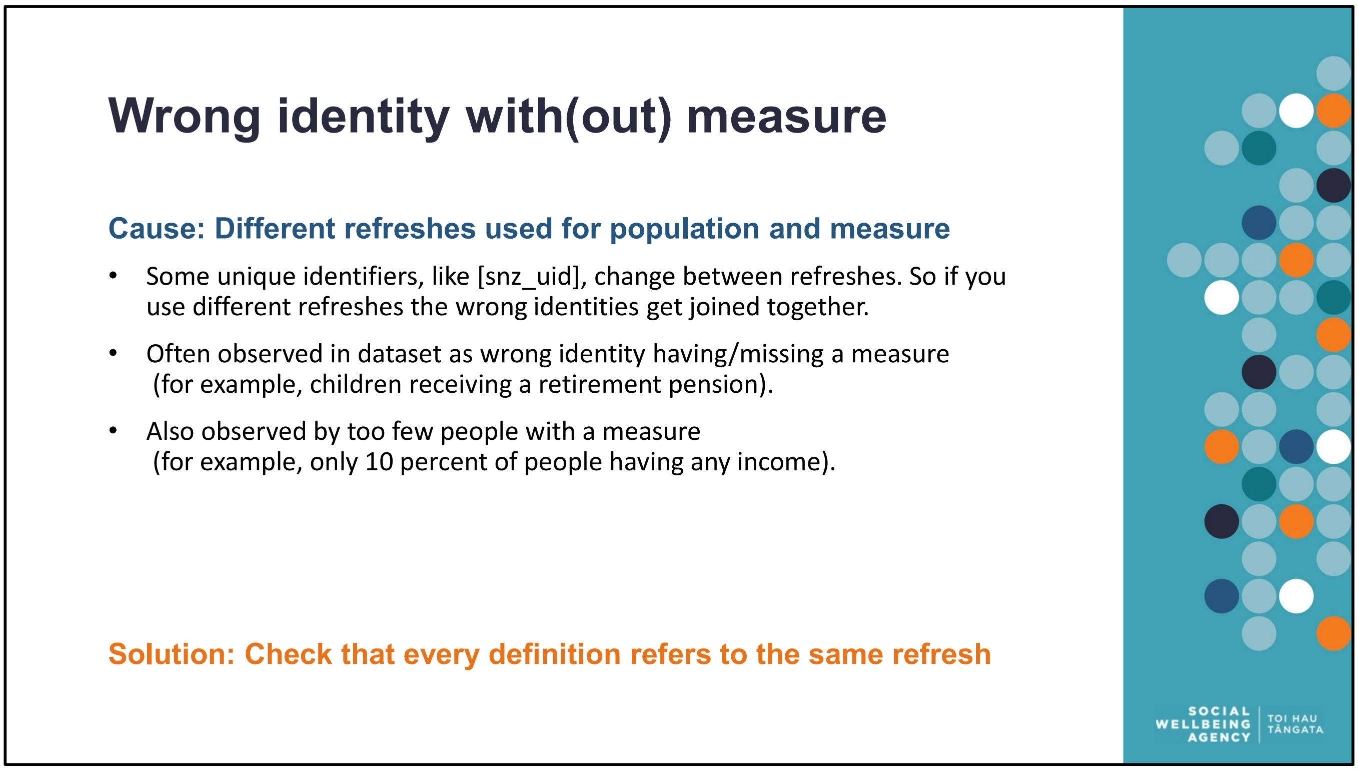
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

DURATION sums up the individual durations of all time periods that are within the study period. It does not check for whether any of the individual durations overlap. This can produce unexpected output such as 400+ days of employment within a single year.

The solution is to merge your overlapping time periods before assembly.

You can find exemplar code for merging spells in the definitions library SWA has published alongside the assembly tool:
https://github.com/nz-social-wellbeing-agency/definitions_library

Good existing definitions to check are employment spells, education spells, or benefit spells. In these SQL definitions we often refer to the merging of overlapping time periods as "Condensing spells".



Wrong identity with(out) measure

Cause: Different refreshes used for population and measure

- Some unique identifiers, like [snz_uid], change between refreshes. So if you use different refreshes the wrong identities get joined together.
- Often observed in dataset as wrong identity having/missing a measure (for example, children receiving a retirement pension).
- Also observed by too few people with a measure (for example, only 10 percent of people having any income).

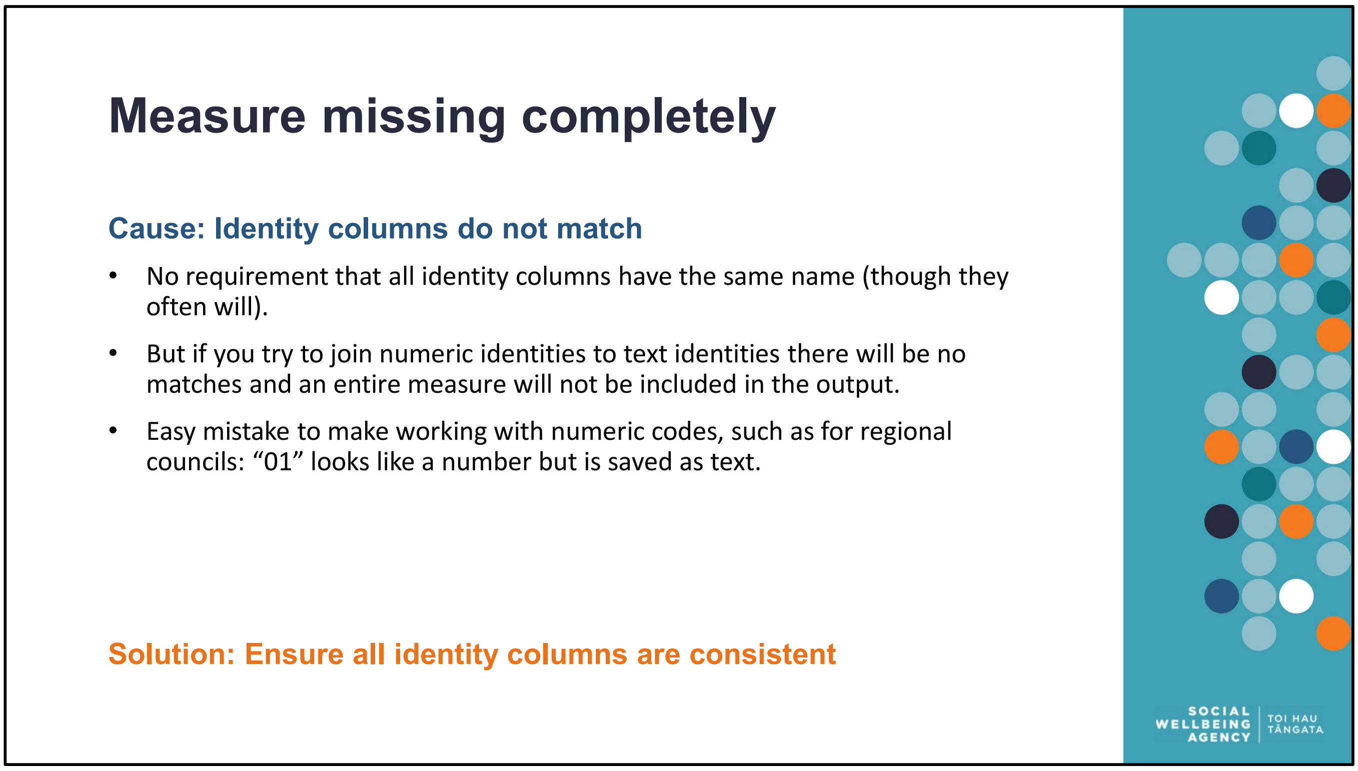
Solution: Check that every definition refers to the same refresh

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Research should be done using a consistent IDI refresh. When some definitions come from one refresh, and other refreshes some from another, then identity codes can change between refreshes.

The best way we currently have to check this is searching all SQL definition files for "IDI_Clean_". Update and rerun any files that are using the wrong refresh.

One supporting tool we hope to see in the future will reduce the risk of this error: A tool that finds the refresh and project schema for every SQL definition. This would make checking for consistency between files much easier.



Measure missing completely

Cause: Identity columns do not match

- No requirement that all identity columns have the same name (though they often will).
- But if you try to join numeric identities to text identities there will be no matches and an entire measure will not be included in the output.
- Easy mistake to make working with numeric codes, such as for regional councils: “01” looks like a number but is saved as text.

Solution: Ensure all identity columns are consistent

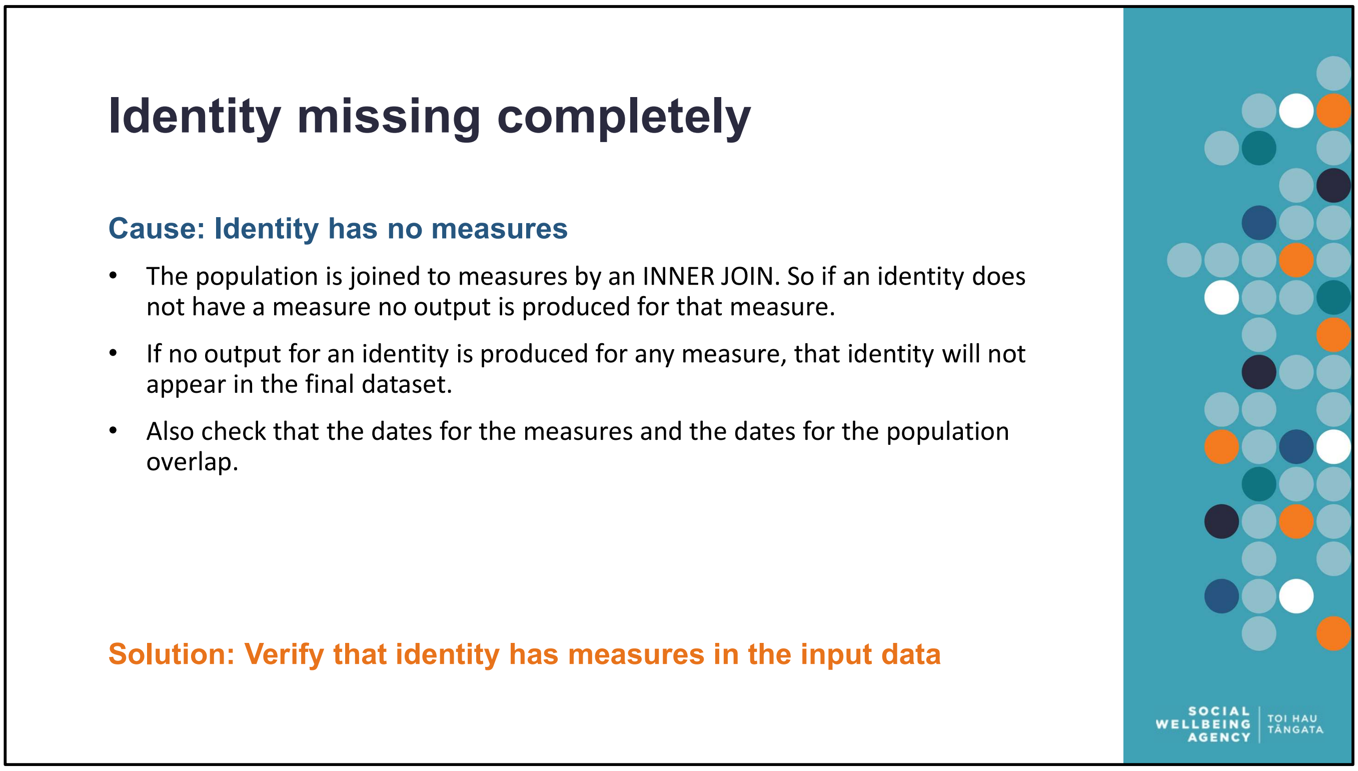
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

The names of identity columns can be different but the format must be the same

- It is critical to ensure that all the identity columns are in same data format.
- Mismatch of datatypes might not always throw error and can lead to incorrect joins.

In some special cases SQL will join numeric and text data. If the identities are smaller numbers (for example regional council codes, which are 2 digits), some codes may join: the text “10” may join to the number 10. But other codes will not join: the text “01” will not join to the number 1.

By default [snz_uid] values are all numeric so this is unlikely to be an problem if using [snz_uid]



Identity missing completely

Cause: Identity has no measures

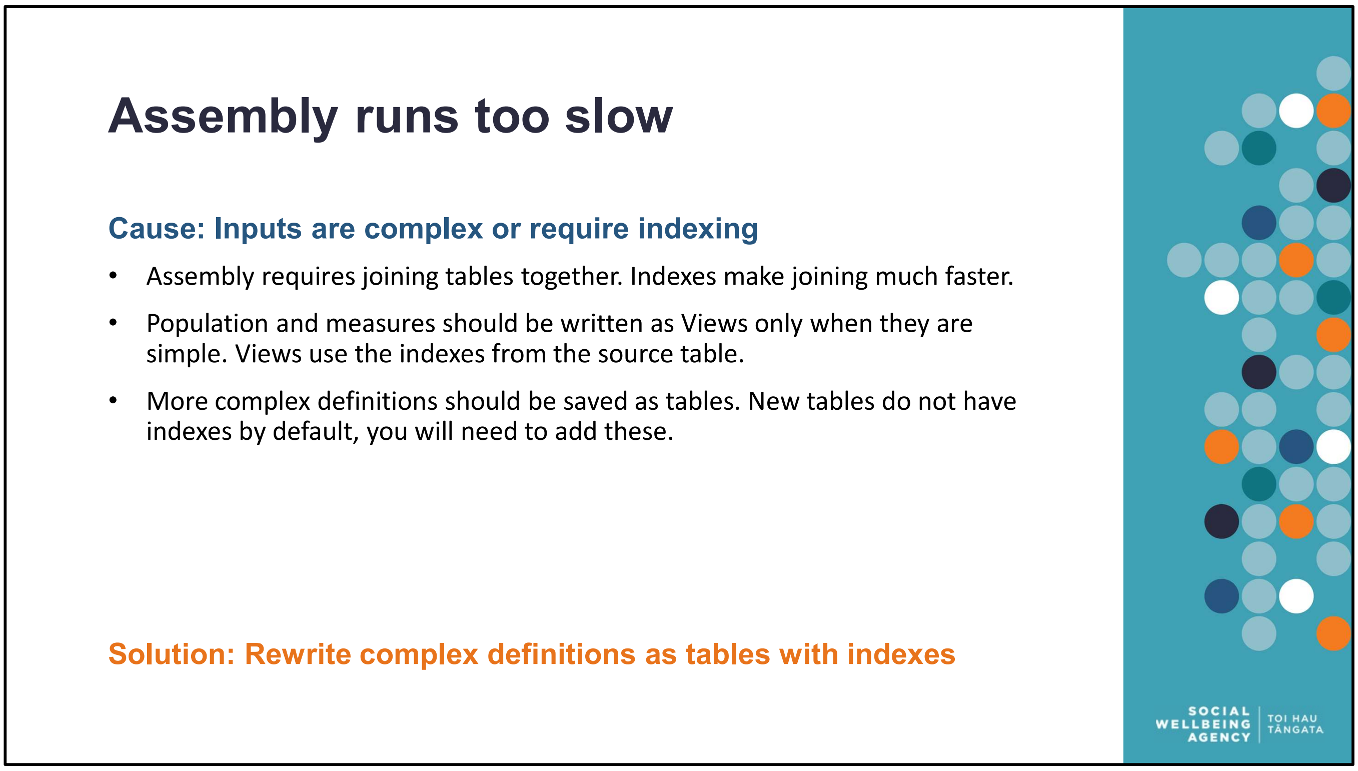
- The population is joined to measures by an INNER JOIN. So if an identity does not have a measure no output is produced for that measure.
- If no output for an identity is produced for any measure, that identity will not appear in the final dataset.
- Also check that the dates for the measures and the dates for the population overlap.

Solution: Verify that identity has measures in the input data

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

If an identity does not have any measures it will not appear in the final output. This is most likely to occur when working with identities that are not linked to the IDI spine, are not in the residential population, or are 'rare' in some way.

If you want to ensure that an identity is included in the output dataset the easiest was is to include the study population in the measure table. Create a new measure that checks if identities in the population table EXISTS in the population table.



Assembly runs too slow

Cause: Inputs are complex or require indexing

- Assembly requires joining tables together. Indexes make joining much faster.
- Population and measures should be written as Views only when they are simple. Views use the indexes from the source table.
- More complex definitions should be saved as tables. New tables do not have indexes by default, you will need to add these.

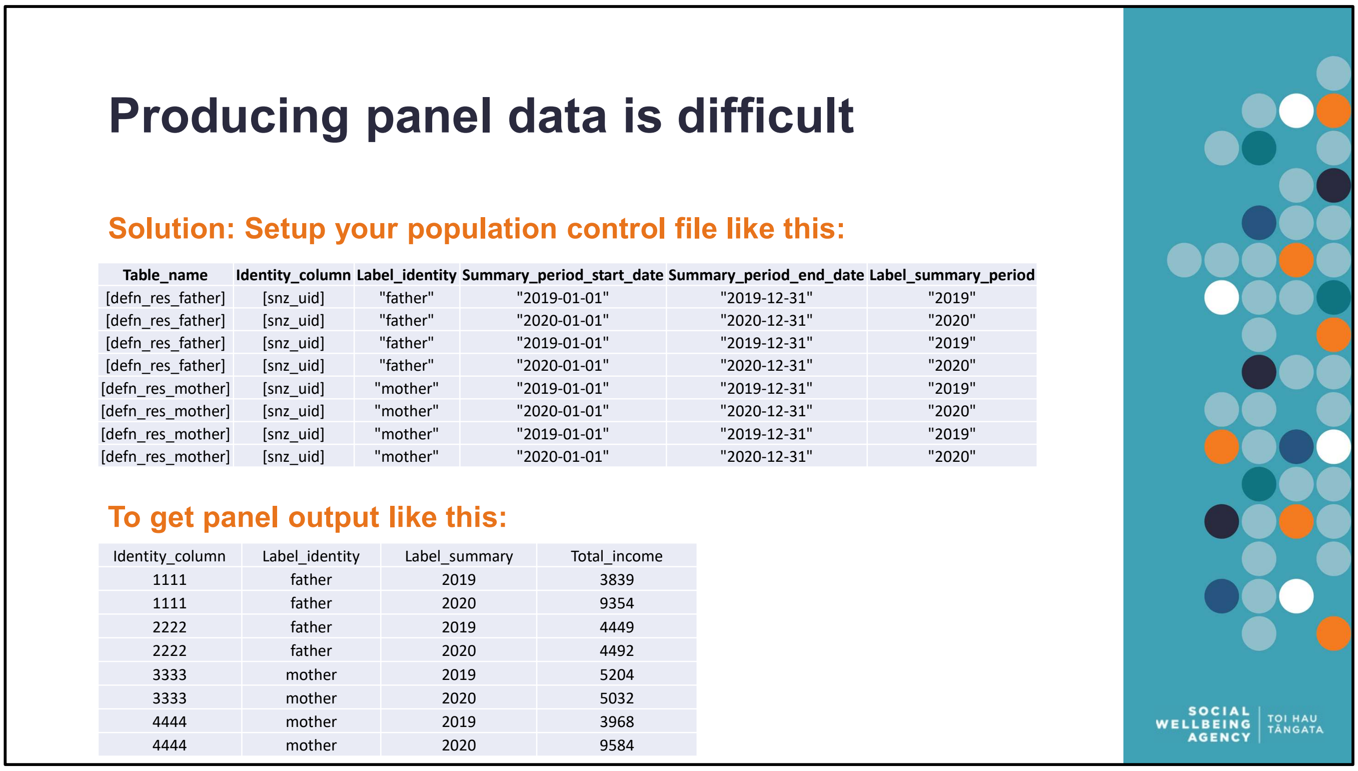
Solution: Rewrite complex definitions as tables with indexes

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

The speed of the assembly tool depends on how fast it is to access and combine the population and measure definitions. Definitions that are slow to access will result in slow performance for the assembly tool.

When running the tool in DEVELOPMENT MODE check for measures that perform slowly. Consider recoding these measures to be more efficient. Adding (clustered) indexes on the identity column improves the assembly speed by a lot.

If you are not sure what definitions are simple and can be Views, and what definitions are too complex and should be tables, take a look at some existing measure definitions. These should also contain code for adding indexes to your tables.



Producing panel data is difficult

Solution: Setup your population control file like this:

Table_name	Identity_column	Label_identity	Summary_period_start_date	Summary_period_end_date	Label_summary_period
[defn_res_father]	[snz_uid]	"father"	"2019-01-01"	"2019-12-31"	"2019"
[defn_res_father]	[snz_uid]	"father"	"2020-01-01"	"2020-12-31"	"2020"
[defn_res_father]	[snz_uid]	"father"	"2019-01-01"	"2019-12-31"	"2019"
[defn_res_father]	[snz_uid]	"father"	"2020-01-01"	"2020-12-31"	"2020"
[defn_res_mother]	[snz_uid]	"mother"	"2019-01-01"	"2019-12-31"	"2019"
[defn_res_mother]	[snz_uid]	"mother"	"2020-01-01"	"2020-12-31"	"2020"
[defn_res_mother]	[snz_uid]	"mother"	"2019-01-01"	"2019-12-31"	"2019"
[defn_res_mother]	[snz_uid]	"mother"	"2020-01-01"	"2020-12-31"	"2020"

To get panel output like this:

Identity_column	Label_identity	Label_summary	Total_income
1111	father	2019	3839
1111	father	2020	9354
2222	father	2019	4449
2222	father	2020	4492
3333	mother	2019	5204
3333	mother	2020	5032
4444	mother	2019	3968
4444	mother	2020	9584

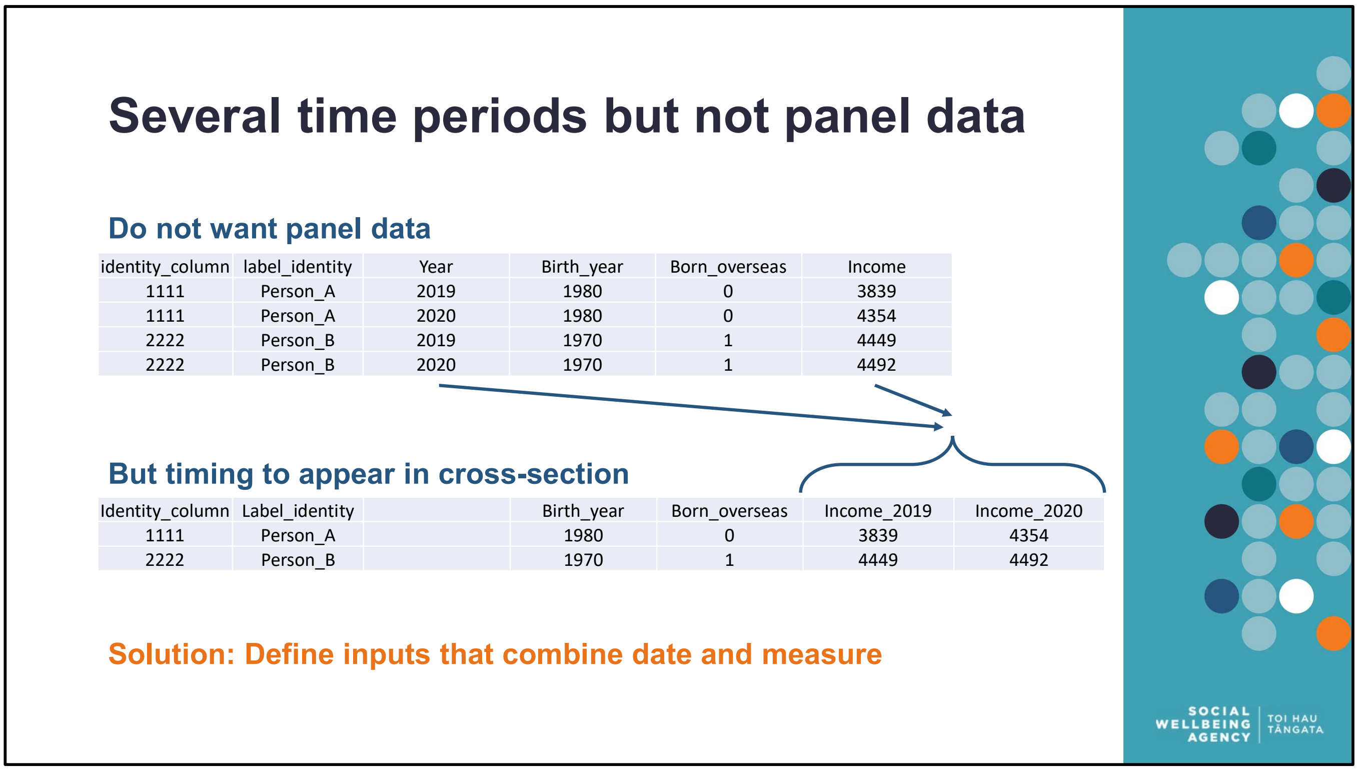
SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Producing panel date is best done by combining identity and summary period labels.

The Label_identity and Label_summary_period inputs makes it easy to repeat the measures for different time-frames and different sub-populations.

For example, if you want to find total income for individuals in 2 set of populations (e.g. fathers and mothers) for 2019 and 2020, you can define your population with two identity labels – “mother” and “father” – and two summary labels – “2019” and “2020”.

You can also use SQL columns for the label identity, start date, end date, and summary period label.
As an alternative to the example above: If you used a single table [defn_res_parents] that contained both mothers and fathers with two columns [snz_uid] and [role], then the population file would only have four rows. Instead of “father” and “mother” in the Label_identity column you would put [role] because the [role] column contains the labels father and mother.



Several time periods but not panel data

Do not want panel data

identity_column	label_identity	Year	Birth_year	Born_overseas	Income
1111	Person_A	2019	1980	0	3839
1111	Person_A	2020	1980	0	4354
2222	Person_B	2019	1970	1	4449
2222	Person_B	2020	1970	1	4492

But timing to appear in cross-section

Identity_column	Label_identity		Birth_year	Born_overseas	Income_2019	Income_2020
1111	Person_A		1980	0	3839	4354
2222	Person_B		1970	1	4449	4492

Solution: Define inputs that combine date and measure

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

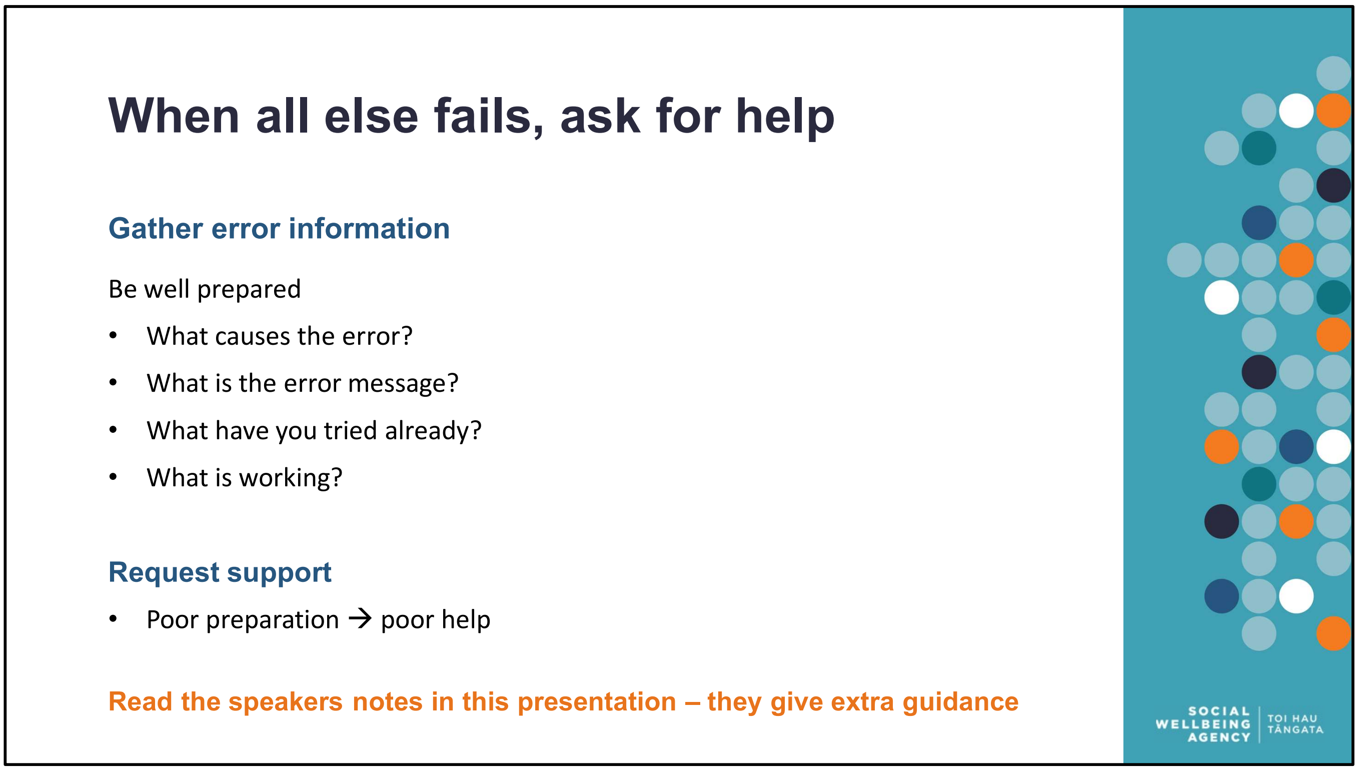
Suppose you want a cross-section dataset but would like income for last year and this year. So rather than two rows per identity with columns [year] and [income] (as a panel dataset would have), you want one row per identity with columns [income_this_year] and [income_last_year].

Several approaches:

1. Assemble a panel and use lag/lead functions to fetch the value for the next year (but this duplicates all the other columns).
2. Assemble two datasets, a master dataset and a minimal dataset for the extra year then join them together (but can not do assembly in one step).
3. Define income for each year in the input measure and include both in the control file (recommended).

For our recommended approach, we can transform a measure table with columns [year] and [income] to a table with columns [year], [income], [income_2019] and [income_2020] by creating two new columns: [income_2019] and [income_2020]. In SQL this might look like:

```
SELECT *
    ,IIF([year] = 2019, [income], NULL) AS [income_2019]
    ,IIF([year] = 2020, [income], NULL) AS [income_2020]
FROM [db].[schema].[table]
```



When all else fails, ask for help

Gather error information

Be well prepared

- What causes the error?
- What is the error message?
- What have you tried already?
- What is working?

Request support

- Poor preparation → poor help

Read the speakers notes in this presentation – they give extra guidance

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Some errors will be due to the assembly tool, and will not be fixable without a developer. The code is open source, so if you have a friendly R developer you could ask them. Otherwise you will need to ask for expert support.

Before you do, make sure you are well prepared.
- Work through this presentation, are there any details you have missed?
- Work through the guidance document
- Isolate the issue – show how it occurs with only one measure and one population.
- Some staff in the Integrated Data team (access2microdata) are familiar with the tool – check with them.

As a final resort, contact the tool developer / maintainer.
Remember, the developer is not a helpdesk, but a fellow researcher.
- How can you help me to help you?

Simon Anastasiadis

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

swa.govt.nz

SOCIAL WELLBEING AGENCY | TOI HAU TĀNGATA

Thank you for your time and attention.

I invite you to adopt the Dataset Assembly Tool for your IDI research.
It has saved me a lot of time and headaches trying to build datasets manually.
There is an initial library of definitions from my use of it.
So I am confident it will add value to your work.