Automatic Conversion of Tables to LongForm Dataframes

by Jimmy Oh

Abstract **TableToLongForm** automatically converts hierarchical Tables intended for a human reader into a simple LongForm dataframe that is machine readable, making it easier to access and use the data for analysis. It does this by recognising positional cues present in the hierarchical Table (which would normally be interpreted visually by the human brain) to decompose, then reconstruct the data into a LongForm dataframe. The article motivates the benefit of such a conversion with an example Table, followed by a short user manual, which includes a comparison between the simple one argument call to `TableToLongForm`, with code for an equivalent manual conversion. The article then explores the types of Tables the package can convert by providing a gallery of all recognised patterns. It finishes with a discussion of available diagnostic methods and future work.

Introduction

TableToLongForm is an R package that automatically converts hierarchical *Tables*[1] intended for a human reader into a *LongForm* R `"data.frame"` that is *machine readable*[2]. While several tools exist that can aid in manipulation of data, such as OpenRefine (OpenRefine, 2013), which can be used to clean messy data, the **speedr** R package (Visne et al., 2012), which can aid in filtering data, and R packages like **reshape2** (Wickham, 2007) and **plyr** (Wickham, 2011), which enable restructuring of the data to focus on specific aspects of the data, for these tools to work their magic you first need machine readable data. However, data released as Tables are *not* machine readable. At best, such tools will provide some aid in manually converting the data to something that is machine readable, a process that is costly in terms of both time and effort. **TableToLongForm** is an automatic tool for converting a family of Tables to a machine readable form, and once so converted the user is free to use their favourite tool, R or otherwise, to make full use of the data.

The article motivates the benefit of such a conversion with an example Table, followed by a short user manual, which includes a comparison between the simple one argument call to `TableToLongForm`, with code for an equivalent manual conversion. The article then explores the types of Tables the package can convert by providing a gallery of all recognised patterns. It finishes with a discussion of available diagnostic methods and future work.

Motivation

There is still a prevalence of data releases being made for direct human consumption in formats that are not machine readable, a significant barrier to effective utilisation of the data. One symptom of this is the release of data in tabular form that relies on a hierarchy that can only be understood after identifying patterns and discerning the structure of the Table, a task easy for a human brain but rather difficult for a computer.

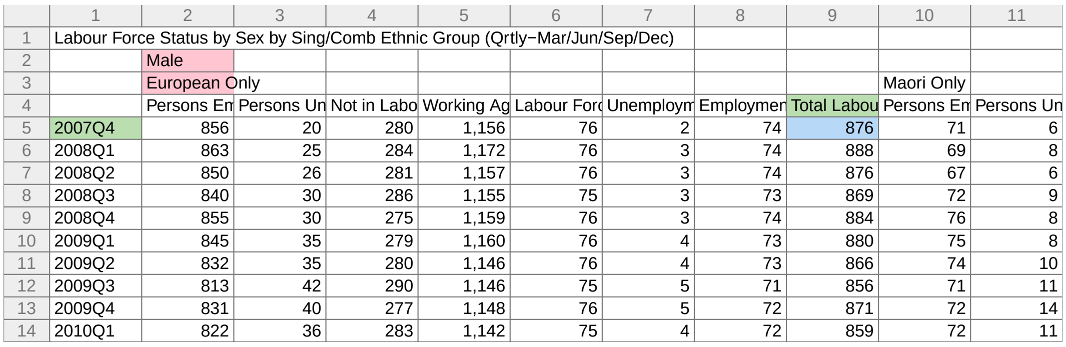
An example of such a Table is shown in Figure 1. For such a Table, the computer will be unable to easily read in the data due to the difficulty in finding all information related to a piece of data. Take the number '`876`' in cell (5, 9) for instance; to collect all the information linked to that number we must look at cell (5, 1) for the time period ('`2007Q4`'), cell (4, 9) for the data heading ('`Total Labour Force`'), cell (3, 2) for the ethnic category ('`European Only`') and cell (2, 2) for the gender category ('`Male`'). Note that, aside from the time period and the data heading, the other information related to cell (5, 9) were neither in the same row nor the same column. The human brain can interpret the positional cues to understand the hierarchy fairly easily, the computer requires a lot more work.

Preparing such data for use would normally require a non-trivial time investment to restructure the data in a manner that can be machine read and used. If such preparatory work was done manually, such work will have to be repeated multiple times as the data is updated. In some cases the data will be spread across multiple files, which means that much more preparatory work. Even if the work is scripted, small changes in the format can easily throw a wrench into the works and break it. All of this quickly adds up to a significant time cost to make use of data released in Tables.

[1] *Table*, with a capital T, is used in this article to specifically mean hierarchical tables, e.g. Figure 1.

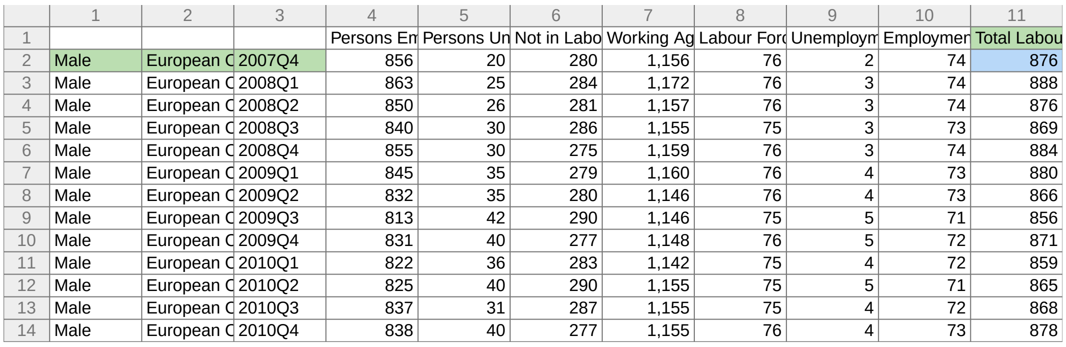
[2] *Machine readable* is used to mean that the format is intended for access and manipulation by computers, and it is thus much easier to use the data for various purposes, such as statistical analysis. It can alternatively be described as *Tidy Data* (Wickham, Submitted), with the conversion taking the data closer to the ideal 'tidy' form.

LongForm is a simple alternative data format that most R users will find familiar as an R `"data.frame"` object, the format that most R functions require of their data. **TableToLongForm** automatically converts Tables to LongForm dataframes[3], which can mean significant savings in time while enabling much greater utilisation of the data. Figure 2 shows Figure 1 after automatic conversion using **TableToLongForm**, which took around a tenth of a second on the author's machine. In particular, note that the same data we examined above, '`876`' now in cell (2, 11), has all related information in the same row (except for the column heading, which is in the same column), making it easy for the computer to understand and manipulate.



	1	2	3	4	5	6	7	8	9	10	11
1	Labour Force Status by Sex by Sing/Comb Ethnic Group (Qrtly−Mar/Jun/Sep/Dec)										
2		Male									
3		European Only								Maori Only	
4		Persons Em	Persons Un	Not in Labo	Working Ag	Labour Forc	Unemploym	Employmen	Total Labou	Persons Em	Persons Un
5	2007Q4	856	20	280	1,156	76	2	74	876	71	6
6	2008Q1	863	25	284	1,172	76	3	74	888	69	8
7	2008Q2	850	26	281	1,157	76	3	74	876	67	6
8	2008Q3	840	30	286	1,155	75	3	73	869	72	9
9	2008Q4	855	30	275	1,159	76	3	74	884	76	8
10	2009Q1	845	35	279	1,160	76	4	73	880	75	8
11	2009Q2	832	35	280	1,146	76	4	73	866	74	10
12	2009Q3	813	42	290	1,146	75	5	71	856	71	11
13	2009Q4	831	40	277	1,148	76	5	72	871	72	14
14	2010Q1	822	36	283	1,142	75	4	72	859	72	11

Figure 1: An example of a hierarchical Table. The Table is of the Labour Force Status data (Statistics New Zealand, 2013) and in total spans 240 columns. The Table is too large to be immediately useful for humans, and cannot even be easily manipulated with a computer, as understanding the data requires linking information across different rows and columns.



	1	2	3	4	5	6	7	8	9	10	11
1				Persons Em	Persons Un	Not in Labo	Working Ag	Labour Forc	Unemploym	Employmen	Total Labou
2	Male	European C	2007Q4	856	20	280	1,156	76	2	74	876
3	Male	European C	2008Q1	863	25	284	1,172	76	3	74	888
4	Male	European C	2008Q2	850	26	281	1,157	76	3	74	876
5	Male	European C	2008Q3	840	30	286	1,155	75	3	73	869
6	Male	European C	2008Q4	855	30	275	1,159	76	3	74	884
7	Male	European C	2009Q1	845	35	279	1,160	76	4	73	880
8	Male	European C	2009Q2	832	35	280	1,146	76	4	73	866
9	Male	European C	2009Q3	813	42	290	1,146	75	5	71	856
10	Male	European C	2009Q4	831	40	277	1,148	76	5	72	871
11	Male	European C	2010Q1	822	36	283	1,142	75	4	72	859
12	Male	European C	2010Q2	825	40	290	1,155	75	5	71	865
13	Male	European C	2010Q3	837	31	287	1,155	75	4	72	868
14	Male	European C	2010Q4	838	40	277	1,155	76	4	73	878

Figure 2: An example of a LongForm dataframe. This is the Table in Figure 1 after automatic conversion with **TableToLongForm** and in total spans 660 rows. Now all related information can be found in the same row or column, making the data much more useful.

User manual

Loading the data

TableToLongForm's preferred argument is a `"matrix"` of mode `"character"`. If a `"data.frame"` is supplied instead, it is coerced to a `"matrix"` with a warning. Empty cells should be classed as `"NA"` for correct operation of the algorithms. Currently **TableToLongForm** does not distinguish between missing values and empty space, both are treated as `"NA"` values.

As the Labour Force Status data used in Figure 1 classifies missing values as '`..`', we must ensure R correctly reads these, in addition to empty cells, as `"NA"` values[4].

```
LabourForce = as.matrix(read.csv("StatsNZLabourForce.csv",
                                  header = FALSE, na.strings = c("", "..")))
```

[3]I use the term LongForm loosely and in some cases **TableToLongForm** will result in WideForm output as the difference can depend on contextual information the computer cannot comprehend. However, the output will be machine readable and many tools, such as those mentioned in the opening paragraph, can be used to further reformat the data, including conversions between LongForm and WideForm.

[4]This Table, after being read in as a `"matrix"` as shown, is included in **TableToLongForm** as part of `data(TCData)`, and can be accessed with `TCData[["StatsNZLabourForce"]]`

Calling TableToLongForm

If the Table can be recognised by **TableToLongForm**, a simple call to `TableToLongForm` with just a single argument is all that is needed. `TableToLongForm` has additional optional arguments used primarily for diagnostic purposes, which are covered in the diagnostics section at the end of the article.

```
LabourForce.converted = TableToLongForm(LabourForce)
```

Aside: manual conversion

For comparison the code for manual conversion of the table is provided below. We note after careful observation of the data that:

- There are 3 gender categories: '`Male`', '`Female`' and '`Total Both Sexes`', each 80 columns in width.
- There are 10 ethnic categories, each a consistent 8 columns in width.
- The data are found in rows 5 to 26.

Armed with this knowledge, we can write the above code that, with some trial and error and cross-checking of results, will successfully convert the Table to a LongForm. This code is fairly compact and efficiency-wise beats **TableToLongForm**, taking a little over a thousandth of a second to make the conversion (compared to about a hundredth of a second for a call to `TableToLongForm`) on the author's machine. However, it took a non-trivial investment of time to code and test the results (it took the author about 30 minutes), is mostly useless for any other Table, and if any of the many strong assumptions it makes are violated (e.g. a new row of data is added), it breaks and requires fixing, which means even more time consumed. All this work and hassle to just *read in the data* in a useful format.

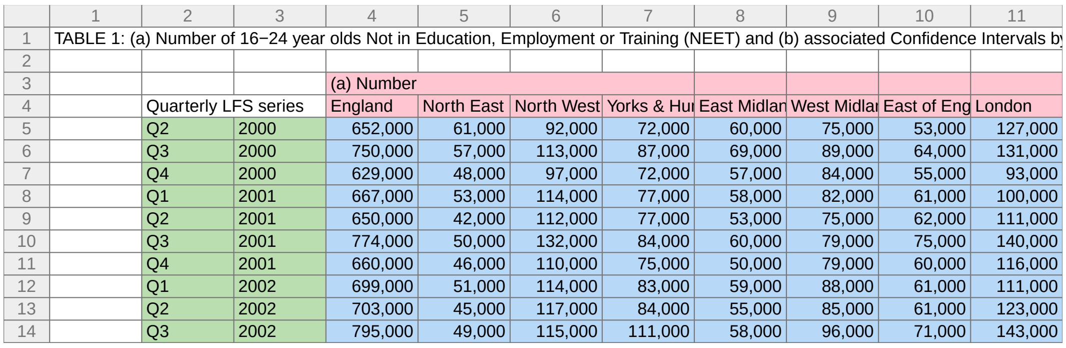
```
LFout = NULL
chYear = LabourForce[5:26, 1]
for(Gender in 0:2)
  for(Ethni in 0:9){
    chGender = LabourForce[2, 2 + Gender * 80]
    chEthni = LabourForce[3, 2 + Ethni * 8]
    LFout = rbind(LFout,
      cbind(chGender, chEthni, chYear,
      LabourForce[5:26, 2 + Gender * 80 + (Ethni * 8):((Ethni + 1) * 8 - 1)])
    )
  }
colnames(LFout) = c("Gender", "Ethnicity", "Time.Period", LabourForce[4, 2:9])
```

IdentResult

For a successful conversion, **TableToLongForm** must first find the Table, that is, it must *Identify* the rows and columns in which the labels and data values can be found. This task can be surprisingly difficult, requiring many fringe-case checks and exception handling. The current core identification algorithm searches for blocks (rectangular regions) of numbers in the supplied `"matrix"`. This region is assumed to contain the data and from it **TableToLongForm** infers the locations of the corresponding labels. The result, after some extra work to handle fringe-cases and the like, is the `IdentResult`, a `"list"` which specifies the rows and columns in which the labels and the data can be found.

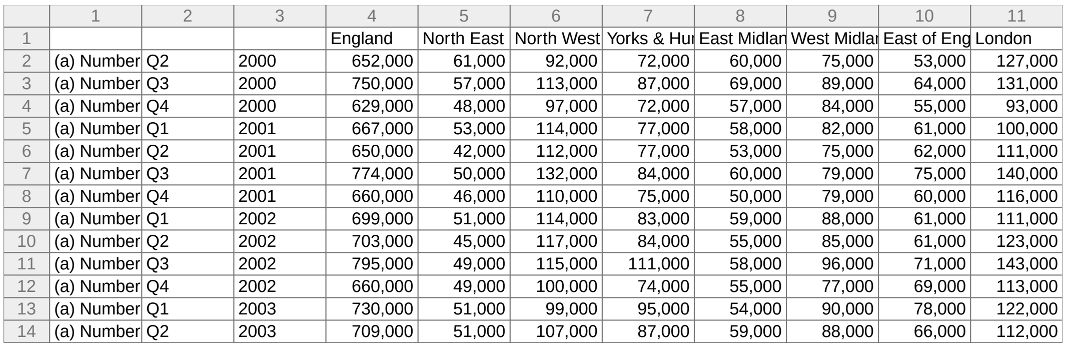
If **TableToLongForm** fails to correctly Identify the correct rows and columns, it is possible to manually specify the `IdentResult` as an argument. This is the case for the Table in Figure 3, where one of the row label columns is a Year column consisting only of numbers. **TableToLongForm**'s numeric label detection algorithm is still quite primitive and fails to correctly identify column 3 as a label, but by manually specifying the `IdentResult`, **TableToLongForm** can still successfully convert the Table; the resulting `"data.frame"` is shown in Figure 4. Even for cases such as this where the `IdentResult` must be manually specified, the work required for the conversion with **TableToLongForm** will be strictly less than for a manual conversion as we would need the same information, and more, to convert manually.

```
TableToLongForm(NEET, IdentResult = list(rows = list(label = 3:4, data = 5:57),
                                         cols = list(label = 2:3, data = 4:24)))
```



	1	2	3	4	5	6	7	8	9	10	11
1	TABLE 1: (a) Number of 16−24 year olds Not in Education, Employment or Training (NEET) and (b) associated Confidence Intervals by										
2											
3				(a) Number							
4		Quarterly LFS series		England	North East	North West	Yorks & Hu	East Midlan	West Midla	East of Eng	London
5		Q2	2000	652,000	61,000	92,000	72,000	60,000	75,000	53,000	127,000
6		Q3	2000	750,000	57,000	113,000	87,000	69,000	89,000	64,000	131,000
7		Q4	2000	629,000	48,000	97,000	72,000	57,000	84,000	55,000	93,000
8		Q1	2001	667,000	53,000	114,000	77,000	58,000	82,000	61,000	100,000
9		Q2	2001	650,000	42,000	112,000	77,000	53,000	75,000	62,000	111,000
10		Q3	2001	774,000	50,000	132,000	84,000	60,000	79,000	75,000	140,000
11		Q4	2001	660,000	46,000	110,000	75,000	50,000	79,000	60,000	116,000
12		Q1	2002	699,000	51,000	114,000	83,000	59,000	88,000	61,000	111,000
13		Q2	2002	703,000	45,000	117,000	84,000	55,000	85,000	61,000	123,000
14		Q3	2002	795,000	49,000	115,000	111,000	58,000	96,000	71,000	143,000

Figure 3: Another example of a hierarchical Table. The Table is of the NEET statistics (Department for Education (UK), 2013) and is relatively tame in terms of complexity. The work required to manually convert and read in such a Table would be light, but still enough to be an annoying time sink. Highlighted are the three regions TableToLongForm must identify for successful conversion, and if automatic identification of these regions fail, the rows and columns corresponding to these three rectangular regions can be specified manually.



	1	2	3	4	5	6	7	8	9	10	11
1				England	North East	North West	Yorks & Hu	East Midlan	West Midla	East of Eng	London
2	(a) Number	Q2	2000	652,000	61,000	92,000	72,000	60,000	75,000	53,000	127,000
3	(a) Number	Q3	2000	750,000	57,000	113,000	87,000	69,000	89,000	64,000	131,000
4	(a) Number	Q4	2000	629,000	48,000	97,000	72,000	57,000	84,000	55,000	93,000
5	(a) Number	Q1	2001	667,000	53,000	114,000	77,000	58,000	82,000	61,000	100,000
6	(a) Number	Q2	2001	650,000	42,000	112,000	77,000	53,000	75,000	62,000	111,000
7	(a) Number	Q3	2001	774,000	50,000	132,000	84,000	60,000	79,000	75,000	140,000
8	(a) Number	Q4	2001	660,000	46,000	110,000	75,000	50,000	79,000	60,000	116,000
9	(a) Number	Q1	2002	699,000	51,000	114,000	83,000	59,000	88,000	61,000	111,000
10	(a) Number	Q2	2002	703,000	45,000	117,000	84,000	55,000	85,000	61,000	123,000
11	(a) Number	Q3	2002	795,000	49,000	115,000	111,000	58,000	96,000	71,000	143,000
12	(a) Number	Q4	2002	660,000	49,000	100,000	74,000	55,000	77,000	69,000	113,000
13	(a) Number	Q1	2003	730,000	51,000	99,000	95,000	54,000	90,000	78,000	122,000
14	(a) Number	Q2	2003	709,000	51,000	107,000	87,000	59,000	88,000	66,000	112,000

Figure 4: Another example of a LongForm dataframe. This is the Table in Figure 3 after automatic conversion with **TableToLongForm**. Although the conversion required the aid of a human to specify the optional argument `IdentResult` to be successful, the work required with **TableToLongForm** will be strictly less than for a manual conversion as we would need the same information, and more, to convert manually.

Recognised patterns

TableToLongForm consists of a number of algorithms that can collectively process a variety of so-called *recognised patterns* of hierarchical structure (also called the *parentage* of the labels). Any Table that consists of some combination of the recognised patterns can be automatically converted with **TableToLongForm**. It is not strictly necessary for a user to know what the patterns are, as they can simply try calling `TableToLongForm` on the Table to see if it converts. All the recognised patterns are listed here for reference purposes[5]. For an example of a real Table that demonstrates a combination of the recognised patterns, refer to **Real Example - NZQA** located at the end of this section.

For each pattern an example table is first shown using toy data, that displays the pattern, followed by a short description of the pattern, and ending with the example table converted with **TableToLongForm**.

Many of the recognised patterns apply only for row labels. Column labels are recognised by noticing that the transpose of column labels can often be processed as row labels, though there are several fringe cases that must be corrected for.

[5]All the Tables demonstrating the recognised patterns are included in **TableToLongForm** as part of `data(TCData)`. `TableToLongForm` can be called on these Tables for the converted versions, e.g. `TableToLongForm(TCData[["ToyExByEmptyBelow"]]`

Empty Below

	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	10	20	30	40
3		Row Child2	11	21	31	41
4	Row Parent2	Row Child1	12	22	32	42
5		Row Child2	13	23	33	43

Above, we have an example of the Empty Below pattern, the most simple type of parentage. Here the *parent* and *children* are in different columns and we can see which of the children belong to which parent through the use of empty space below each parent. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	10	20	30	40
3	Row Parent1	Row Child2	11	21	31	41
4	Row Parent2	Row Child1	12	22	32	42
5	Row Parent2	Row Child2	13	23	33	43

Empty Right 1

	1	2	3	4	5	6	7
1				Column 1	Column 2	Column 3	Column 4
2	Row Parent1			10	20	30	40
3	Row Child1	Row Child−Child1		11	21	31	41
4	Row Child2	Row Child−Child2		12	22	32	42
5	Row Parent2			13	23	33	43
6	Row Child1	Row Child−Child1		14	24	34	44
7		Row Child−Child2		15	25	35	45

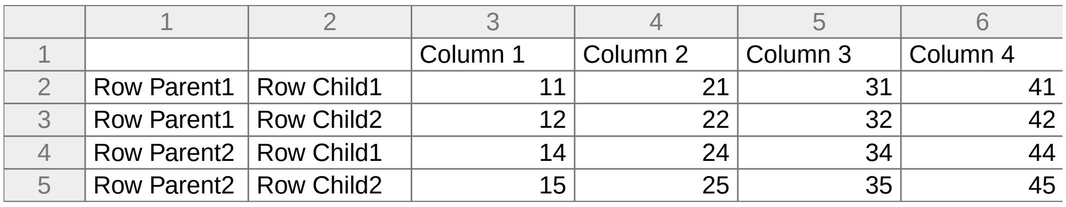
Above, we have an example of the most basic form of the Empty Right pattern. In this situation we have children in the same column as their parent. We can still recognise these as children if the children have children (*Child-Child*) in a different column, while the parent does not (and hence the parent is Empty Right). Note the values pertaining to the Parent (if any) are discarded. This is because they are assumed to simply represent the sum of their children's values. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7
1				Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	Row Child−Ch	11	21	31	41
3	Row Parent1	Row Child2	Row Child−Ch	12	22	32	42
4	Row Parent2	Row Child1	Row Child−Ch	14	24	34	44
5	Row Parent2	Row Child1	Row Child−Ch	15	25	35	45

Empty Right 2

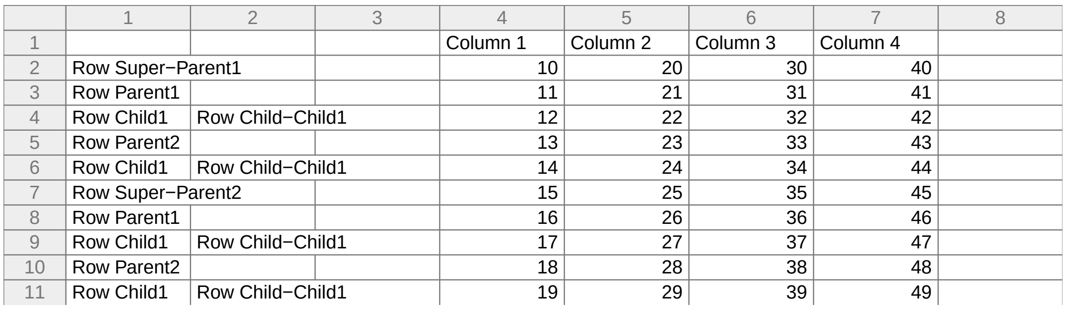
	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1		10	20	30	40
3		Row Child1	11	21	31	41
4		Row Child2	12	22	32	42
5	Row Parent2		13	23	33	43
6		Row Child1	14	24	34	44
7		Row Child2	15	25	35	45

Above, we have an example of both Empty Below and Empty Right. Either algorithm can handle this situation, but simply due to the ordering of the algorithms such situations are handled as Empty Right. The Table after conversion to a LongForm follows.



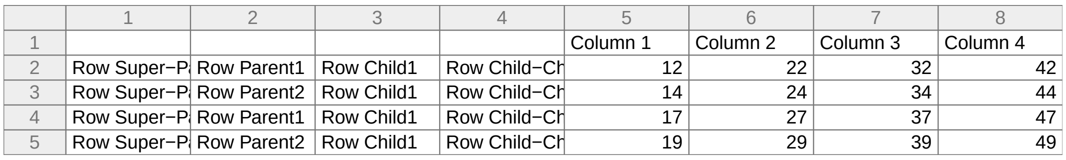
	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	11	21	31	41
3	Row Parent1	Row Child2	12	22	32	42
4	Row Parent2	Row Child1	14	24	34	44
5	Row Parent2	Row Child2	15	25	35	45

Empty Right 3



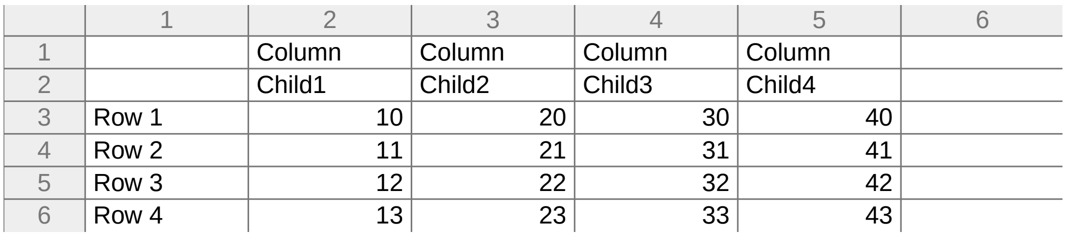
	1	2	3	4	5	6	7	8
1				Column 1	Column 2	Column 3	Column 4	
2	Row Super−Parent1			10	20	30	40	
3	Row Parent1			11	21	31	41	
4	Row Child1	Row Child−Child1		12	22	32	42	
5	Row Parent2			13	23	33	43	
6	Row Child1	Row Child−Child1		14	24	34	44	
7	Row Super−Parent2			15	25	35	45	
8	Row Parent1			16	26	36	46	
9	Row Child1	Row Child−Child1		17	27	37	47	
10	Row Parent2			18	28	38	48	
11	Row Child1	Row Child−Child1		19	29	39	49	

Above, we have an example of a complex version of the Empty Right pattern. The "parent-child in the same column" situation has been extended further and we now have parents (*Super-Parent*) who have children (*Parent*), who each further have children (*Child*), all in the same column. Such situations can still be recognised if the lowest-level children in the column (*Child*) have children in a different column (*Child-Child*), while its direct parents (*Parent*) each have children in the same column (*Child*) but not in a different column (is Empty Right), and the top-most parents (*Super-Parents*) also have no children in a different column (is also Empty Right). The algorithm cannot currently handle super-super-parents. The Table after conversion to a LongForm follows.



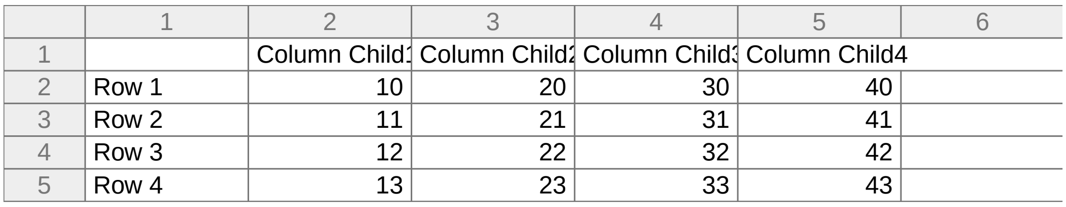
	1	2	3	4	5	6	7	8
1					Column 1	Column 2	Column 3	Column 4
2	Row Super−P	Row Parent1	Row Child1	Row Child−Ch	12	22	32	42
3	Row Super−P	Row Parent2	Row Child1	Row Child−Ch	14	24	34	44
4	Row Super−P	Row Parent1	Row Child1	Row Child−Ch	17	27	37	47
5	Row Super−P	Row Parent2	Row Child1	Row Child−Ch	19	29	39	49

Multi-row Column Label



	1	2	3	4	5	6
1		Column	Column	Column	Column	
2		Child1	Child2	Child3	Child4	
3	Row 1	10	20	30	40	
4	Row 2	11	21	31	41	
5	Row 3	12	22	32	42	
6	Row 4	13	23	33	43	

Above, we have an example of Multi-row Column Labels. Often column labels are physically split over multiple rows rather than making use of line breaks in the same cell. In such occurrences, any row not identified as a parent are collapsed into a single row of labels. The Table after conversion to a LongForm follows.



	1	2	3	4	5	6
1		Column Child1	Column Child2	Column Child3	Column Child4	
2	Row 1	10	20	30	40	
3	Row 2	11	21	31	41	
4	Row 3	12	22	32	42	
5	Row 4	13	23	33	43	

Mismatched Column Label

	1	2	3	4	5	6	7	8	9
1		Col Child1		Col Child2		Col Child3		Col Child4	
2	Row 1		10		20		30		40
3	Row 2		11		21		31		41
4	Row 3		12		22		32		42
5	Row 4		13		23		33		43

Above, we have an example of Mismatched Column Labels. Sometimes the column labels are in a different column to the data, usually due to a misguided attempt at visual alignment of labels to the data. As long as the correct rows and columns were identified for the data and the labels (see User manual subsection on `IdentResult`), and if there are the same number of data columns as label columns, these mismatched column labels will be paired with the data columns. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8	9
1		Col Child1	Col Child2	Col Child3	Col Child4				
2	Row 1	10	20	30	40				
3	Row 2	11	21	31	41				
4	Row 3	12	22	32	42				
5	Row 4	13	23	33	43				

Misaligned Column Label

	1	2	3	4	5	6	7	8	9
1			Col Parent1				Col Parent2		
2		Col Child1	Col Child2	Col Child3	Col Child4	Col Child1	Col Child2	Col Child3	Col Child4
3	Row 1	10	20	30	40	50	60	70	80
4	Row 2	11	21	31	41	51	61	71	81
5	Row 3	12	22	32	42	52	62	72	82
6	Row 4	13	23	33	43	53	63	73	83

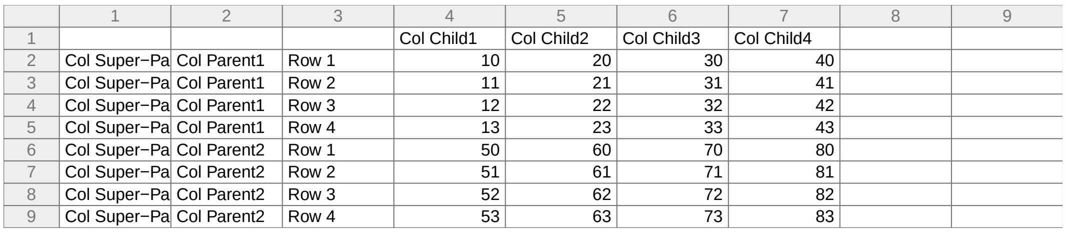
Above, we have an example of Misaligned Column Labels. Often column parents are physically centred over their children (N.B. where a spreadsheet's cell-merge feature is used to do the centering, the actual value is usually stored in the top-left cell and hence causes no problems). **TableToLongForm** makes use of pattern recognition to identify repeating patterns in the labels, or in empty cells surrounding the labels, to correct for the misalignment. For the *Column Parents* row, we find (starting from column 2, the first data column) a pattern of Empty-NonEmpty-Empty-Empty, with the pattern occurring twice. In the *Col Child* row, we also find a pattern of length 4 occurring twice. This can be used to correctly align the *Column Parents* to its children. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8	9
1			Col Child1	Col Child2	Col Child3	Col Child4			
2	Col Parent1	Row 1	10	20	30	40			
3	Col Parent1	Row 2	11	21	31	41			
4	Col Parent1	Row 3	12	22	32	42			
5	Col Parent1	Row 4	13	23	33	43			
6	Col Parent2	Row 1	50	60	70	80			
7	Col Parent2	Row 2	51	61	71	81			
8	Col Parent2	Row 3	52	62	72	82			
9	Col Parent2	Row 4	53	63	73	83			

Misaligned Column Label 2

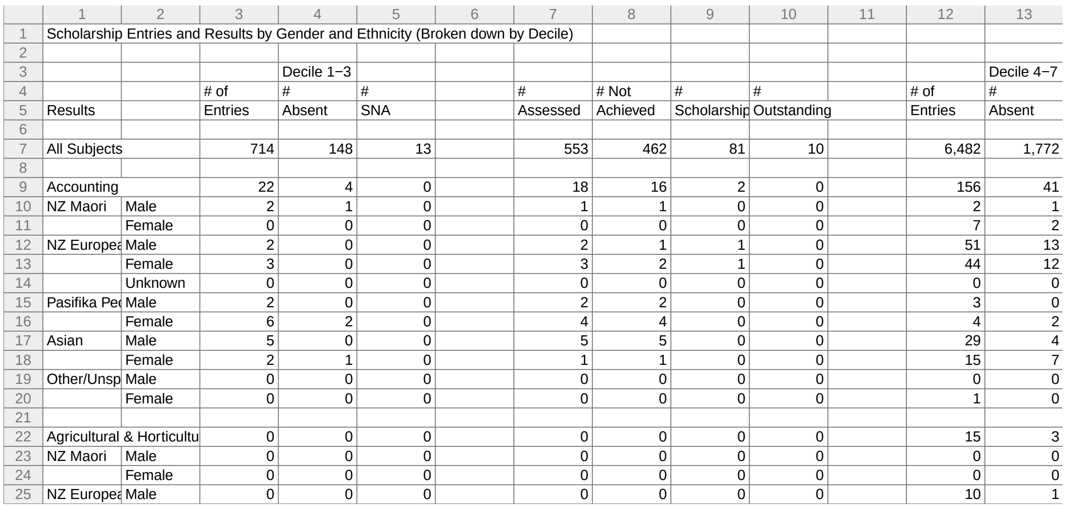
	1	2	3	4	5	6	7	8	9
1				Col Super−Parent					
2			Col Parent1				Col Parent2		
3		Col Child1	Col Child2	Col Child3	Col Child4	Col Child1	Col Child2	Col Child3	Col Child4
4	Row 1	10	20	30	40	50	60	70	80
5	Row 2	11	21	31	41	51	61	71	81
6	Row 3	12	22	32	42	52	62	72	82
7	Row 4	13	23	33	43	53	63	73	83

Above, we have a generalised example of Misaligned Column Labels. We now have *Column Super-Parent* which is misaligned to both its direct children, the *Column Parents*, and to the lowest-level children. The Table after conversion to a LongForm follows.



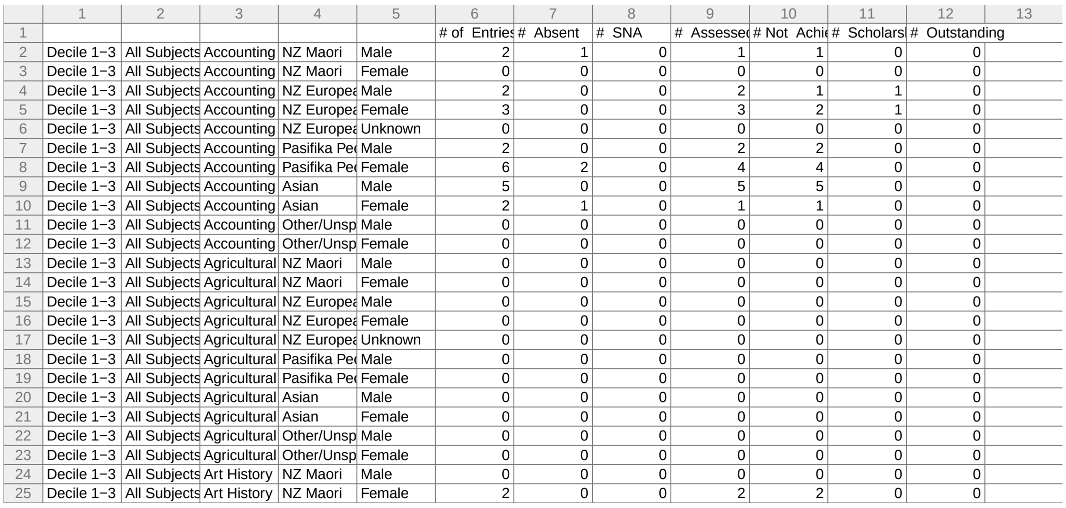
	1	2	3	4	5	6	7	8	9
1				Col Child1	Col Child2	Col Child3	Col Child4		
2	Col Super−Pa	Col Parent1	Row 1	10	20	30	40		
3	Col Super−Pa	Col Parent1	Row 2	11	21	31	41		
4	Col Super−Pa	Col Parent1	Row 3	12	22	32	42		
5	Col Super−Pa	Col Parent1	Row 4	13	23	33	43		
6	Col Super−Pa	Col Parent2	Row 1	50	60	70	80		
7	Col Super−Pa	Col Parent2	Row 2	51	61	71	81		
8	Col Super−Pa	Col Parent2	Row 3	52	62	72	82		
9	Col Super−Pa	Col Parent2	Row 4	53	63	73	83		

Real Example - NZQA



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	Scholarship Entries and Results by Gender and Ethnicity (Broken down by Decile)												
2													
3				Decile 1−3									Decile 4−7
4			# of	#	#		#	# Not	#	#		# of	#
5	Results		Entries	Absent	SNA		Assessed	Achieved	Scholarship	Outstanding		Entries	Absent
6													
7	All Subjects		714	148	13		553	462	81	10		6,482	1,772
8													
9	Accounting		22	4	0		18	16	2	0		156	41
10	NZ Maori	Male	2	1	0		1	1	0	0		2	1
11		Female	0	0	0		0	0	0	0		7	2
12	NZ Europea	Male	2	0	0		2	1	1	0		51	13
13		Female	3	0	0		3	2	1	0		44	12
14		Unknown	0	0	0		0	0	0	0		0	0
15	Pasifika Pe	Male	2	0	0		2	2	0	0		3	0
16		Female	6	2	0		4	4	0	0		4	2
17	Asian	Male	5	0	0		5	5	0	0		29	4
18		Female	2	1	0		1	1	0	0		15	7
19	Other/Unsp	Male	0	0	0		0	0	0	0		0	0
20		Female	0	0	0		0	0	0	0		1	0
21													
22	Agricultural & Horticultu		0	0	0		0	0	0	0		15	3
23	NZ Maori	Male	0	0	0		0	0	0	0		0	0
24		Female	0	0	0		0	0	0	0		0	0
25	NZ Europea	Male	0	0	0		0	0	0	0		10	1

Above, we have an example of real data released in a Table[6] that demonstrates a combination of many of the patterns listed above. The data comes from the New Zealand Qualifications Authority (NZQA, 2012) regarding the entries and results for their Scholarship Exams. The Table demonstrates Empty Below, Empty Right (Including Type 3, as '`All Subjects`' is a *super-parent*), Multi-row Column Labels and Misaligned Column Labels. The Table is substantially more complex than the Labour Force Status data used in Figure 1, and will require considerably more work to convert manually. Worse, each year of the data is stored in a separate Table, each with slight differences in format. Thus the manual conversion code would have to either be individually tweaked for each year (requiring yet more work), or be flexible enough to handle these differences (requiring substantially more work). Other data from NZQA faces bigger problems; though the Scholarships data can all be obtained in a mere 8 Tables (for 8 years from 2004 to 2011), the Subjects data not only requires a separate Table for each year, but also for each subject (of which there are 91). Obtaining the greatest breakdown possible for the Subjects data across all other variables requires thousands of individual Tables. Without automatic conversion with **TableToLongForm**, simply reading in such data for use would require too much work to be practical. The Table after conversion to a LongForm follows.



	1	2	3	4	5	6	7	8	9	10	11	12	13
1						# of Entries	# Absent	# SNA	# Assesse	# Not Achi	# Scholars	# Outstanding	
2	Decile 1−3	All Subjects	Accounting	NZ Maori	Male	2	1	0	1	1	0	0	
3	Decile 1−3	All Subjects	Accounting	NZ Maori	Female	0	0	0	0	0	0	0	
4	Decile 1−3	All Subjects	Accounting	NZ Europea	Male	2	0	0	2	1	1	0	
5	Decile 1−3	All Subjects	Accounting	NZ Europea	Female	3	0	0	3	2	1	0	
6	Decile 1−3	All Subjects	Accounting	NZ Europea	Unknown	0	0	0	0	0	0	0	
7	Decile 1−3	All Subjects	Accounting	Pasifika Pe	Male	2	0	0	2	2	0	0	
8	Decile 1−3	All Subjects	Accounting	Pasifika Pe	Female	6	2	0	4	4	0	0	
9	Decile 1−3	All Subjects	Accounting	Asian	Male	5	0	0	5	5	0	0	
10	Decile 1−3	All Subjects	Accounting	Asian	Female	2	1	0	1	1	0	0	
11	Decile 1−3	All Subjects	Accounting	Other/Unsp	Male	0	0	0	0	0	0	0	
12	Decile 1−3	All Subjects	Accounting	Other/Unsp	Female	0	0	0	0	0	0	0	
13	Decile 1−3	All Subjects	Agricultural	NZ Maori	Male	0	0	0	0	0	0	0	
14	Decile 1−3	All Subjects	Agricultural	NZ Maori	Female	0	0	0	0	0	0	0	
15	Decile 1−3	All Subjects	Agricultural	NZ Europea	Male	0	0	0	0	0	0	0	
16	Decile 1−3	All Subjects	Agricultural	NZ Europea	Female	0	0	0	0	0	0	0	
17	Decile 1−3	All Subjects	Agricultural	NZ Europea	Unknown	0	0	0	0	0	0	0	
18	Decile 1−3	All Subjects	Agricultural	Pasifika Pe	Male	0	0	0	0	0	0	0	
19	Decile 1−3	All Subjects	Agricultural	Pasifika Pe	Female	0	0	0	0	0	0	0	
20	Decile 1−3	All Subjects	Agricultural	Asian	Male	0	0	0	0	0	0	0	
21	Decile 1−3	All Subjects	Agricultural	Asian	Female	0	0	0	0	0	0	0	
22	Decile 1−3	All Subjects	Agricultural	Other/Unsp	Male	0	0	0	0	0	0	0	
23	Decile 1−3	All Subjects	Agricultural	Other/Unsp	Female	0	0	0	0	0	0	0	
24	Decile 1−3	All Subjects	Art History	NZ Maori	Male	0	0	0	0	0	0	0	
25	Decile 1−3	All Subjects	Art History	NZ Maori	Female	2	0	0	2	2	0	0	

[6]This Table is included in **TableToLongForm** as part of `data(TCData)`, and can be accessed with `TCData[["NZQAScholarships"]]`

Diagnostics

The primary limitation of **TableToLongForm** is that the function will be a black box to most users. After running the function on a Table, the user will either be given back a "`data.frame`" with no easy way of verifying if the result is correct, or be confronted with an error with little idea of what went wrong. Based on ad hoc tests conducted so far, **TableToLongForm** will either succeed, or fail catastrophically in a manner that is easily recognised as utter failure. However methods for verifying correct operation (or to understand failures) would be desirable.

The simplest method currently available is to examine the additional output returned when `TableToLongForm` is called with the optional argument `fulloutput = TRUE`. This will return the 'final product' of **TableToLongForm**'s algorithms in the form of '`IdentResult`', '`rowplist`' and '`colplist`'.

IdentResult was covered above in the user manual section and contains information on where the data and labels are found.

rowsplist and colsplist stand for Row/Column Parentage List which are nested "`list`" objects that represents all the hierarchical relationships in the Table, as identified by **TableToLongForm**. For easier reading they are assigned the "`plist`" class which has a custom '`print`' method. An example of a '`colplist`' is shown in Figure 5.

```
> TableToLongForm(LabourForce, fulloutput = TRUE)[["colplist"]]
+ Male (1, 2)
- + European Only (1, 3)
- - + Persons Employed in Labour Force (1, 4)
- - + Persons Unemployed in Labour Force (2, 4)
- - + Not in Labour Force (3, 4)
- - + Working Age Population (4, 4)
- - + Labour Force Participation Rate (5, 4)
- - + Unemployment Rate (6, 4)
- - + Employment Rate (7, 4)
- - + Total Labour Force (8, 4)
- + Maori Only (9, 3)
- - + Persons Employed in Labour Force (9, 4)
## Output truncated
```

Figure 5: A truncated example of the '`colplist`' for the Labour Force Status data used in Figure 1. It represents the hierarchical relationships of the column labels, as identified by **TableToLongForm**. We can see that it has correctly identified '`Male`' as a top-level parent with the ethnic categories, such as '`European Only`', nested inside, which are in turn parents to the lowest-level categories, such as '`Employment Rate`'.

Unfortunately this output has two key limitations. First, it is not obvious from this output what went wrong (or if nothing went wrong), requiring some detective work to piece together the evidence. Second, if anything did go wrong, the user still does not know *why*.

The option with the potential to provide the most information is calling `TableToLongForm` with the optional argument `diagnostics = TRUE`, which will write diagnostic output to a file, printing key variables at each major stage of the conversion process. This output can thus be used to track **TableToLongForm**'s progress as it works to convert the Table, enabling fairly accurate assessment of where exactly it went wrong. Some example diagnostic output is shown in Figure 6. Unfortunately understanding this output requires familiarity with the workings of the code and is unlikely to be of much use to anyone other than the author.

Discussion

This article has introduced **TableToLongForm**, an R package that can automatically convert hierarchical Tables that would normally rely on the discerning powers of a human brain, to a simple LongForm dataframe that any decent software package can easily manipulate and use. While it can handle a variety of Tables automatically, and an even greater variety with some aid from the human user, it is not without limitations. Ultimately, **TableToLongForm** still uses algorithms to detect a known set of recognised patterns and any Table that deviates from these patterns will break **TableToLongForm**.

There is work to be done in refining and generalising existing algorithms and creating new algorithms so that **TableToLongForm** can successfully handle more cases, while also reducing the

```
###TCR CIMCB rowData
[1]  5 26
###TCR CIMCB colData
[1]   2 241
###TCR IOOC plist
$rows
[1] 1 2 3 4 5 6 7 8
$cols
[1] 4
```

```
###TCR IOOC res
+ Persons Employed in Labour Force (1, 4)
+ Persons Unemployed in Labour Force (2, 4)
+ Not in Labour Force (3, 4)
+ Working Age Population (4, 4)
+ Labour Force Participation Rate (5, 4)
+ Unemployment Rate (6, 4)
+ Employment Rate (7, 4)
+ Total Labour Force (8, 4)
```

Figure 6: A few examples of the diagnostic output generated by `TableToLongForm` when called with '`diagnostics = TRUE`' on the Labour Force Status data used in Figure 1. The diagnostic output is printing key variables at each major stage of the conversion process. '`###TCR`' indicates an identifier line, the following word indicates the part of the function generating this output (e.g. '`CIMCB`', which is short for Call Identification algorithm Most-Common-Boundary), the last word indicates the name of the variable being printed (e.g. '`rowData`'). The diagnostic output can be used to see what **TableToLongForm** is getting right (or wrong)... assuming the user is familiar with the code.

possibility of a false positive. These range from adding more robust checks to the algorithms to verify correct detection, such as sum-checks or pattern recognition, to more fundamental changes, such as altering the `"NA"` classification to distinguish between empty space and missing values. A recent addition to the package (introduced in version 1.3.0) is to enable new, custom algorithms to be used in place of the default ones included with the package. More information on this can be found in the official webpage for the package.

There is also work to be done in diagnostics output, not only in the formal diagnostic output, but also in the form of error and warning messages. Consider for instance the following error message if we call `TableToLongForm` on the Table in Figure 3 without specifying the correct `IdentResult`. From these messages it is not at all obvious that the problem is an incorrect `IdentResult`, which is a problem that is relatively easy to address if only it can correctly be identified by the user.

```
Error in 1:ncol(datbit) : argument of length 0
In addition: Warning message:
In rbind(matColLabel[!fullrows, , drop = FALSE], collapsedlabels) :
  number of columns of result is not a multiple of vector length (arg 2)
```

In terms of formal diagnostic output, various ideas are being tried such as graphical representations of the information provided by `fulloutput = TRUE` by drawing the original Table with the regions being highlighted in some way. Such a method would, for example, make it easier to see a problem with `IdentResult`, as it should become apparent on the drawn Table that the incorrect regions are being highlighted.

This article has been written for **TableToLongForm** version 1.3.1. The code for reproducing the figures in this article, as well as more detailed documentation on the code itself, can be found at https://www.stat.auckland.ac.nz/~joh024/Research/TableToLongForm/. The development version can be found on github at https://github.com/joh024/TableToLongForm.

Bibliography

Department for Education (UK). NEET statistics quarterly brief: April to June, 2013. URL https://www.gov.uk/government/publications/neet-statistics-quarterly-brief-april-to-june-2013. [p4]

NZQA. Secondary School Statistics, 2012. URL http://www.nzqa.govt.nz/. [p8]

OpenRefine. OpenRefine, 2013. URL http://openrefine.org/. [p1]

Statistics New Zealand. Infoshare, 2013. URL http://www.stats.govt.nz/infoshare/. [p2]

I. Visne, A. Yildiz, E. Dilaveroglu, K. Vierlinger, C. Nöhammer, F. Leisch, and A. Kriegner. speedR: An R package for interactive data import, filtering and ready-to-use code generation. *Journal of Statistical Software*, 51(2):1–12, 2012. URL http://www.jstatsoft.org/v51/i02/. [p1]

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL http://www.jstatsoft.org/v21/i12/. [p1]

H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1): 1–29, 2011. URL http://www.jstatsoft.org/v40/i01/. [p1]

H. Wickham. Tidy data. *The Journal of Statistical Software*, Submitted. [p1]

Jimmy Oh
Department of Statistics
The University of Auckland
New Zealand joh024@aucklanduni.ac.nz