Democratising Survey Data with **iNZight**

Tom Elliott[1,2], Chris Wild[2], Andrew Sporle[3,2]

1. Te Rourou Tātaritanga, Victoria University of Auckland
2. Department of Statistics, University of Auckland
3. iNZight Analytics

Abstract

Survey data is often full of unique and complex features that must be dealt with before the data can be analysed. This adds many additional levels of complexity to a valuable data resource. By providing a set of processes for describing complex survey designs and metadata about variables, **iNZight**

1 Introduction

iNZight (Elliott et al., inproc) is . . .

- GUI for data analysis and visualisation
- for learning, and lowering barrier to entry
- can be used by researchers low on money/time/skill
- however still issues around survey data - complex sampling design, additional caveats with coding, etc

2 Importing complex survey designs

- complex survey designs with stratification/clustering - users need to know/remember design structure
- replicate weight designs - users need to specify all (sometimes 100) weights, know/remember the replicate weight method (BRR, JKN, ...)

- also post-stratification (calibration by known values)
- **iNZight** features way of importing this information from a *survey specification* file, distributed with the survey data
- novice users (population researchers, etc) can easily import the survey without worrying about getting the design wrong - then "business as usual" for making graphs and tables

3 Coded variables in surveys

- another common feature is coded variables - often the decoding is held separately in a "metadata" file
- users need to first convert integer variables to factors, and then specify the levels
- for R users, this can typically be coded and run easily
- in a graphical user interface (GUI) like **iNZight**, it's a time-consuming, repetitive process
- however, **iNZight** understands its own "metadata" format (prepended to comma separated values (CSV) files) describing variables in the data
- e.g., 1=Male, 2=Female
- this is common enough - but in some surveys there are additional problems
- missing-value codes in a numeric variables (e.g., refused to answer vs don't know)
- often coded with values higher than the data (e.g., if variable takes values 0-20, missing values might be coded with 88, 99)
- this skews the results if not accounted for - so **iNZight** metadata can specify these values, and when the data loads they are automatically converted to `NA`
- if labels are specified ('Don't Know' vs 'Refused to answer') a new variable is created to keep this info

- finally, some survey questions can take multiple responses (e.g., 'What heating methods do you have in your home?')
- typically, all answers would be coded in individual binary variables
- however, it is also common enough practice to record all responses in a single field, using a delimiter: e.g., `1;2;5;8`
- not only do these need decoding, but also splitting into a binary response variable so they can be visualised using existing software (e.g., iNZightMR, Elliott et al., 2020b).

4 Reading survey data into iNZight

- here's a demo survey dataset
- to read this into R, we'd need to do the following:

```
R> library(survey)
R> sample_data <- read.csv('data/sample.csv')
R> sample_data$sex <- factor(sample_data$sex, labels = c('male', 'female'))
R> sample_svy <- svydesign(~cluster, weights = ~wt,
+      strata = ~strata, data = sample_data)
R> svyby(~height, ~sex, sample_svy, svymean)
```

```
          sex   height        se
male     male 165.5385 9.2147687
female female 169.2647 0.8869419
```

iNZight uses the `smart_read()` function from the **iNZightTools** (Elliott et al., 2020a) package to import data, which detects file type and parses meta-data automatically at the top of a file.

```
R> library(iNZightTools)
R> sample_svy <- import_survey('data/sample.svydesign')
```

So we'll need to fall back on using non-overleaf for this one :)

From other paper:

4.0.1 Metadata for delimited files

One issue with delimited data formats (for example CSV) is lack of coding information about variables. For example, a vector with values 1, 2, 3, and 4 might be an integer, however it may also be a coded *factor* variable. Typically, users would have to convert this variable to a factor ("categorical" in **iNZight**) and then label the levels, which would be found in additional documentation supplied with the data. However, this is a lot to ask of novice users, and when there are 10s or even 100s of coded factor variables in a dataset, it becomes very tedious in a GUI system.

To work around this problem, we have developed a *metadata* system for supplying and implementing additional information. Currently, this comes in a very naïve format where information is added to the top of a file, but future endeavours will allow **iNZight** to read metadata from a separate file. As an example, say a variable called `var1` contains coded levels 1–4, with labels 'a', 'b', 'c', and 'd'. The metadata line below is added to the top of the CSV file:

```
#' @factor var1[a,b,c,d]
```

When the file is read in, **iNZight** automatically parses this metadata and imports the variable called `var1` as a factor with the appropriate labels.

The metadata support goes further, allowing renaming of levels, variable names, and even supplying missing-value codes in numeric-variables. The latter is common in surveys, in which 'missing value' responses may be coded with values such as 88 for "Don't know" and 99 for "Refused to answer". By providing this information in the metadata, users avoid producing graphs and summaries of the numeric variable including the coding missing values in the results.

Acknowledgements

This work was conducted by Te Rourou Tātaritanga (`https://terourou.org`), a research group at Victoria University of Wellington funded by an MBIE Endeavour Grant, ref 62506 ENDRP. Recent **iNZight** development has also been supported by iNZight Analytics.

References

T. Elliott, O. Jin, Y. He, and D. Barnett. ***iNZightTools**: Tools for **iNZight***, 2020a. Available from `https://cran.r-project.org/package=iNZightTools`.

T. Elliott, J. Zeng, and S. Potter. ***iNZightMR**: Tools for Exploring Multiple Response Data*, 2020b. URL `https://inzight.nz`. R package version 2.2.5.

T. Elliott, C. Wild, D. Barnett, and A. Sporle. **iNZight**: A graphical user inferface for data visualisation and analysis through R. inproc.