iNZight: a GUI for Learning Statistics

Tom Elliott and Chris Wild and Daniel Barnett and Andrew Sporle

November 30, 2020

Abstract

Getting started with data science is a daunting task, particularly when it requires a large amount of coding before you can even start looking at data. graphical user interfaces (GUIs) have often been used as a way of proving novice users the ability to interact with complex systems without the need for coding. However, many of these themselves have steep learning curves to understand how to make the software do what's needed, and do not provide a pathway to more standard and flexible methods, such as coding. iNZight is a GUI based tool written in R that provides students of statistics and data science the opportunity to interact with data and explore without first learning to code. The tool is designed to be easy to use, with logical interactions and clever defaults. However, it also provides some more complex features to manipulate and analyse data, and further provides a code history of the actions performed, creating a pathway between GUI and learning to code for those interested in progressing into the more open and exciting world of data science.

1 Introduction

- Scope out the need for iNZight

- R can be daunting for beginners/students (who may never have used or even see code before)
- Excel (Microsoft Corporation, 2018), SPSS/etc are rather complex with a not-insignificant learning curve to be able to produce basic exploratative plots and summary statistics
  - Do not provide any kind of a 'pathway' to learning R for data science purposes, either
- Other R-based GUIs:
  - Rcommander (Fox, 2005)
  - Jamovi (The jamovi project, 2020)
  - ???
- Other tools targeting students:
  - NZGrapher (Wills, 2020) uses PHP
- No obvious—simple—point-and-click interfaces for simple data analysis/visualisation that also provide a pathway to more complex, code-driven analyses

2 A history of iNZight

- Originally a simple implementation experimenting with R (R Core Team, 2020b) and 'gWidgets' (now superceeded by 'gWidgets2', (Verzani, 2019)) for making graphs which react to the type of data (i.e., the user doesn't have to choose the graph type)

- Uses GTK (The GTK+ Team, 2020) to produce graphical interface, accessed via the 'RGtk2' R package (Lawrence and Temple Lang, 2010)
- The software uses the variable types (numeric or categorical) to determine the type of graph or summary produced
- Picked up by [. . . ?] and rolled out for use in NCEA Level 3 statistics in New Zealand (final year of high school)
- Redesigned in 2014 with gWidgets2 and reference classes (one of R's Object Oriented Programming approaches)
- Additionally uses a suite of complimentary R packages to separate form (the UI) from function (data processing, graphics, etc)
- Additionally modules for time series, model fitting, etc, and more recently added an add-on system
- Most of the work has been student-driven: "By students, for students" (rather than being created by computer scientists)

3 An overview of iNZight's structure

Producing cross-platform GUIs has always been a difficult task as different operating systems (OSs) implement different display devices. Therefore many projects have been created in an attempt to make cross-platform applications a possibility. One such example is GTK+, which is implemented on Windows, macOS, and Linux systems, providing a single toolkit for creating GUIs for all major systems (The GTK+ Team, 2020).

Of course, interfacing with such a framework is in itself a difficult job, and requires some complex C++ coding. Fortunately, several interfacing packages have been written in R (R Core Team, 2020b) which prove a simple,

platform (and indeed toolkit) independent application programming interface (API) for writing GUIs from R. The 'RGtk2' package (Lawrence and Temple Lang, 2010) provides an platform-independent interface between R and GTK+2, allowing access to most of the classes to construct a GUI that reacts to a user's input. Additionally, the 'gWidgets2' package (Verzani, 2019) provides a framework-independent inteface between R and several other R packages responsible to creating GUIs, namely 'gWidgets2RGtk2' for communicating with 'RGtk2' (Verzani, 2020). Together, these packages make it possible for any R programming to construct a graphical application without any knowledge of GTK or platform-specific development. Indeed, it is the combination of these tools which made it possible for statistics students to create and work on 'iNZight'.

Given a platform- and framework-independent API, the next critical step is planning the internal structure of the application, most importantly ensuring that future development will not become hindered by early decisions. The most foundational decision in the early development of 'iNZight' was to separate *form* from *function*: that is, the code the controls the interface should be, as far as possible, separate from the code that handles data processing, graphics, and so on. Further, we wanted the individual components of the GUI to be independent to ensure future development would be easier: for example, buttons can be moved and replaced, and new components can be added or old ones removed without affecting anything else. The necessity for object oriented programming (OOP) was clear, so that each individual component is represented by a single *class*, which could be modified independently of others, or modified ("inheritance") to make similar widgets with common behaviours but several unique features.

Within the R system there are several OOP implementations, however we

chose to use *reference classes* as these are part of base R and also used by 'gWidgets2'. Each component of the 'iNZight' interface is defined using one or more reference class objects, each of which has a set of *fields* and *methods*. Fields describe properties associated with the objects, most importantly user-specified values (for example an Import window might have a field for the file name). Methods are functions which carry out actions, and have access to the object's fields (the Import window might have a method to load the data, for example).

To separate form from function, methods that perform actions on the data or generate output for the user call functions in other packages. The interface allows users to specify values for the object's fields, and either automatically or by clicking a button call the methods which compile the fields and pass them to an external function. For example, an Import data window might take the file chosen by the user and pass it to `iNZightTools::smart_read()`, a function in the 'iNZightTools' package (Elliott, 2020) which acts as a wrapper for several other data import functions. In this way, the interface does not need to know anything about the data type it is loading: it only collects information from the user and passes it to another function. An early advantage of this was the creation of 'iNZight Lite', an 'shiny'-based alternative to the desktop version of 'iNZight' which can run in a user's browser. In this case, both of the GUIs collect information from the user—potentially in a different way—and pass it to the external function. This way, the result returned (given the same input) will be the same across implementations of 'iNZight'.

A further advantage of the code-separation is that it provides the opportunity of a "stepping stone" between using the GUI and coding directly in R using specialised packages and functions. Rather than learning all of the different data import functions in a range of packages (such as 'readr' (Wickham et al., 2018),

Table 1: iNZight R package family

Package	Description
iNZight	The main package for the GUI
iNZightModules	An additional GUI package providing additional modules for the main 'iNZight' program.
iNZightPlots	Provides plot function `inzplot()` along with `inzsummary()` for descriptive statistics and `inzinference()` for inference and hypothesis testing.
iNZightRegression	Plots and summaries of regression models, including from `lm`, `glm`, and `svyglm` objects.
iNZightTS	Time series visualisation, decomposition, and forecasting.
iNZightMR	Visualisation and estimation of multiple response data.
iNZightTools	A suite of helper functions for data process and variable manipulation.

'readxl' (Wickham and Bryan, 2019), or 'foreign' (R Core Team, 2020a)) they need only learn one function `smart_read()` to import CSV, Excel, SPSS, Stata, or SAS files.

- examples:
  - `iNZightTools::smart_read()` imports a dataset based on its extension - user doesn't need to know `read_csv()`, `read_dta()`, etc
  - `iNZightPlots::inplot()` is the main power-house function within iNZight - takes UI inputs and generates a graph based on the variable types (and other selections)
  - both of these functions can be accessed directly from R
  - most also return the 'tidyverse' code so learners can get a taste for the actual code necessary to do stuff (e.g., filtering data, etc)

iNZight's code separation has led to an entire family of R packages, which are displayed in table 1. Most of these packages can be used standalone, and provide simple wrappers for commonly used R workflows, replicating many of the behaviours in *R for Data Science* (Wickham and Grolemund, 2017). Further, most of the functions return the 'tidyverse' (Wickham et al., 2019) code

used behind-the-scences, providing a further stepping stone for users to learn these more complicated code workflows.

Returning to the underlying structure, we now get a glimpse of the importance of those early decisions on the future development prospects of 'iNZight'. Using reference classes allows us to add and alter individual components without affecting others, and simultaneously providing a singular interface with an external function. Changes to those functions are automatically inherited by 'iNZight', such as storing of the underlying code. Since the data structure is also itself a reference class object, it can apply methods when a change to the data is triggered, namely looking for attached code and appending that to the code history widget.

4 Features of iNZight

At its heart, iNZight is a data visualisation and exploration tool for those users with little to no prior experience with data science or statistics, and who lack the programming demands of more mainstream tools such as Python (Van Rossum and Drake, 2009) and R (R Core Team, 2020b). Therefore, many of the main features relate to exploring data through visualisation, with some data manipulation techniques built in, including specification of survey designs which are automatically incoporated into the rest of iNZight. Since many users will likely want to move on to coding, and since iNZight is built with R, we provide the code history for actions the user makes.

4.1 Data wrangling

The first thing most users will want to do is import their data. iNZight provides an easy to use *Import Data* window which uses the file extension to detect the file type and provide a preview of the data in the same window. This allows

users to quickly see if everything is OK and, if necessary, adjust some of the type-specific options to get it correct. An example of this might be reading european CSV files, which use a semi-colon delimiter instead of a comma.

Once loaded, iNZight provides several important data operations, allowing users to reshape, filter, and otherwise transform their dataset. Many of these 'workflows' are taken from "R for Data Science" (Wickham and Grolemund, 2017). These basic dataset operations are implemented using packages from the 'tidyverse' (Wickham et al., 2019). For each, the GUI provides an interface with inputs corresponding to various arguments, generating an R code call which is evaluated and stored in the script. In some cases, a preview of the resulting dataset is provided, making it easier for users to investigate the result of different options. FIGUREX shows the reshape data window, allowing users to convert from wide format to long format, which is more useful for plotting.

- uses 'tidyverse' methods and workflows to perform some data transformation
- all are calls to wrappers inside the 'iNZightTools' package
- the interface allows users to manipulate arguments to the wrapper function - reactive in some cases (i.e., inputs appear/disappear or change values based on previous inputs)
- in many cases, a preview is displayed to help with what can be complex actions, e.g.

As well as the dataset operations are *variable manipulations*, allowing users to modify individual variables in the data. For example, simple transformations (log, square-root) to renaming or reordering levels of a categorical variable, each has its own interface window that interfaces with 'tidyverse' code to perform

the operations. And, if the operation you want is not available, you can specify a custom command to create a new variable.

The goal of these features is to allow users to import a range of data sets in a range of formats and convert them into a form useful for plotting—that is, *tidy format* (Wickham and Grolemund, 2017), where each row contains a single set of observations about an individual.

4.2 Graphics and simple data analyses

The foremost tool in iNZight's inventory is graphics, which are chosen automatically based on the users's chosen variables. For example, a numeric variable is displayed to the user as a dot plot or, if there are more than 5000˜observations, a histogram, without the user needing to choose this first. A factor (refered to within iNZight as *categorical*) shows as a bar graph. This means the user focusses on exploring the data without the need to first *understand* the data.

In many other data analysis programs, graphs are created by the user first selecting the *type* of graph to display, and then choosing the variable. In an explorative sense, this makes little sense, as for example a variable called "age" might be numeric *or* categorical (for example age groups). The basic types of graphs available in 'iNZight' are shown in table 2. **??**tab:inzplotlarge; shows "large sample" alternatives which are used when sample sizes exceed 5000.

- all powered by 'iNZightPlots' functions
  - `inzplot()` for graphics
  - `inzsummary()` for simple summary information
  - `inzinference()` for inferencial information and hypothesis testing
- uses the variable type(s) to choose the graph type. Figure x shows the available graph types

Table 2: iNZight default plot types are determined by the variable type of the first variable and, if specified, the type of the second variable.

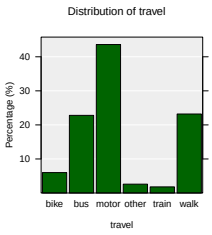
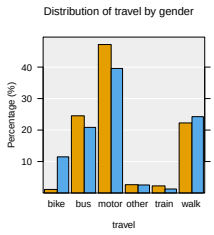
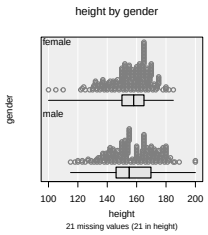
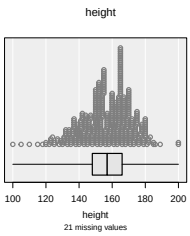
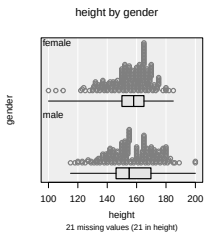
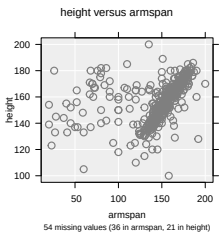
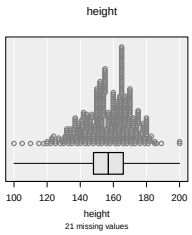
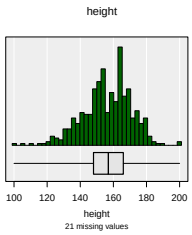
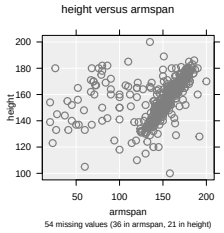
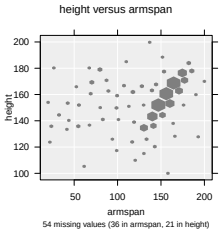
Variable 1	Variable 2		
	None	Categorical	Numeric
Categorical			
Numeric			

Table 3: iNZight's alternative plots for large samples. Other plot types do not have a large sample alternative (i.e., bar charts).

Plot	Small	Large
dot plot		
scatter plot		

- subsetting by 1 or 2 variables is possible/easy/emphasized
  - includes a slider to cycle or 'step' between levels (can add 'motion' to graphs; (Rosling, 2010))
- plot features: specific to plot type, users only see what's relevant
- some features (such as colour by) might apply to different plot types (scatter and dot plot, for example) and so selection is retained when switching between these plot types
- inferencial markup of plots
  - normal theory error bars/curves
  - or bootstrap alternative
  - "Comparison intervals" are a way of visually assessing differences - only shown on differences *that should be compared* - bar plot example
  - an approximate tukey thing? details needed here ('iNZightMR' package)
- summary information
  - a simple numeric summary *of the current plot* (philosophy: look first)
- inference information
  - again, default inferences *of the current plot*
  - both normal and bootstrap options
  - gives a list of relevant hypothesis tests (cf. other software which requires you to decide on hypothesis test first)

4.3 Saving code history

- many functions (the wrappers) generate code and attach it to their result

```
data.filtered <- iNZightTools::filterData(iris, "Species", "
  Setosa")
cat(code(data.filtered))
# iris %>% dplyr::filter(Species == "Setosa")

```

- the iNZight GUI stores the attached code after each action and appends it to the R history 'script'
- provides a record of what was done
- helps students interested in continuing data science/statistics to get familiar with code before writing it themselves (can copy+paste and modify, for example)

4.4 Analysing surveys with iNZight

- survey data:
  - iNZight gets told once, and everything else 'just works'
  - plots use survey methods and alternative plot types (histogram vs dotplot; hex bin plot vs scatter plot)
  - summary/inference all use survey methods to obtain summaries/inferences/hypothesis tests
- supports strata/cluster based survey designs, replicate weights, and post-stratification

4.5 Other modules

- A suite of other (fixed) modules for analying/exploring special types of data
  - time series
  - multiple response
  - maps
- or for doing specific tasks
  - model fitting
- each is its own "reference class" object connecting to one (or more) functions in another package
  - time series -> 'iNZightTS'
  - multiple response -> 'iNZightMR'
  - maps -> 'iNZightMaps'
  - model fitting -> 'iNZightRegression'
- these are mostly wrappers to other functions, or modified versions -> made to work as 'standalone' packages (i.e., without iNZight)
- not fully implemented, but making progress towards modules also code-writing
- also a newer add-on system allowing anyone to write new modules for iNZight (added manually or through our github-hosted repository)
- easier/faster to maintain/update
- useful for e.g., teachers of a specific course

5 Distributing and running iNZight

Deploying software is a difficult process, particularly when your users use a range of operating systems. Fortunately, since iNZight is essentially just a collection of R packages, it's very simple to install since R itself is already distributed for the most popular operating systems. However, many of our users are students with little software knowledge, so a simpler method is necessary. Note also that some of our R packages are not available on CRAN, so we need a way for those to be available also.

The first issue is to make the R packages we develop available online. The first choice would be CRAN, and at the time of writing we have [four or five] packages accepted. The main GUI packages 'iNZight' and 'iNZightModules', however, require a significant amount of work to ensure they adhere to the CRAN repository policies: they mostly center around reading and writing to the disk and saving of users' preferences. An alternative is to self-host the packages, which has two advantages:

1. we can upload packages instantly, and
2. updates and bug fixes can be released frequently, whereas CRAN suggests limiting updates to every 2–3 months.

We have a cloud-based repository using Amazon Web Services (AWS) at `https://r.docker.stat.auckland.ac.nz`. The "docker" component of the URL is there only for historic reasons, and will one day be updated.

Given packages available (details later) we have a distribution system available for Windows allowing users to install iNZight as a standalone program. This works because, on Windows, R can be installed to any directory of choice, and runs without requiring access to any other part of the filesystem. Therefore we bundle the R program, the R package library (including iNZight R pack-

age and its dependencies) and several other files to make it possible to launch iNZight via R with a simple double-click.

The build process requires compilation of all the package binaries for Windows, structuring of the iNZightVIT folder, building of the EXE installer, and finally deployment to the server for users to download. GitHub Actions (`https://github.com/features/actions`) is a useful automation framework which we now use to automate the entire build and deploy process for 'iNZight'. Using several individual *jobs*, the automation builds the latest package binaries and uploads them—along with the source files—to our R package repository. Then it builds the windows install directory with these binaries, and uses NSIS (NSIS Team and NSIS Community, 2020) to create an executable windows installer, which is uploaded to the remote server.

Obviously, the CRAN releases are managed manually and less frequently.

5.1 A note for macOS and Linux users

iNZight uses GTK as the window management system, however in recent years support for GTK on macOS has been neglected, and so we can no longer provide an official release of iNZight for Mac users. It is, however, possible to compile the necessary package dependencies manually and get it running, but this is not something we expect our users to do.

On Linux, support varies by distribution. Ubuntu, Debian, and Fedora-based systems natively support R, and GTK binaries are available, so it is easy enough to install the dependencies and R packages. However, to help, we provide a simple GNU Make script to install iNZight and create launch scripts, which may be installed onto the system to make `inzight` available from anywhere.

We do provide an online version of iNZight, called iNZight Lite, that provides much of the same functionality but runs in the user's web browser and connects

to a remote R server.

- ✓GTK support for mac is dead - so we cannot support macOS any longer
- ✓Linux requires installation of some platform-specific packages (gtk, xorg, etc etc)
- ✓just like installing any other R package
- ✓but we do include a build tool to help install and set-up run/update scripts to make launching easier

5.2 Manual installation

Installing iNZight manually is usually a simple process. On Windows, no dependencies are required beforehand; on macOS and Linux, users must first install GTK and various other platform-dependent dependencies. And R of course. Then it's a matter of starting R and installing from our repository:

```
install.packages('iNZight',
    dependencies = TRUE,
    repos = c('https://r.docker.stat.auckland.ac.nz',
        'https://cran.rstudio.com'),
    Ncpus = 4L # recommended on Linux
)
```

Afterwards, iNZight can be run by loading the library and invoking the function of the same name:

```
library(iNZight)
iNZight()
# optionally pass a dataset directly:
iNZight(iris)
```

6 Discussion and Future Work

- iNZight is a simple-to-use tool for exploring/visualising/analysing data
- built for beginners with capabilities to progress to R-driven coding
- easy-to-learn makes it ideal for organisations that need to do specific jobs rarely (i.e., every few months) - more complicated tools are easy to forget
- flexibility of design makes it easy to add/change functionality as demand changes/grows (e.g., survey design)

6.1 Future Work

- better survey handling
- filling the 'code writing' gaps
- connecting to databases (and potentially processing as much as possible within the db)
- Te Reo translation (and flexibility to add more)
- ...

Acronyms

API application programming interface. 4

AWS Amazon Web Services. 14

GUI graphical user interface. 1, 3–5, 8, 14

OOP object oriented programming. 4

OS operating system. 3

References

Elliott, T. (2020). *iNZightTools: Tools for 'iNZight'*. Available from `https://cran.r-project.org/package=iNZightTools`.

Fox, J. (2005). The R Commander: A basic statistics graphical user interface to R. *Journal of Statistical Software*, 14(9):1–42.

Lawrence, M. and Temple Lang, D. (2010). RGtk2: A graphical user interface toolkit for R. *Journal of Statistical Software*, 37(8):1–52.

Microsoft Corporation (2018). *Microsoft Excel*. `https://office.microsoft.com/excel`.

NSIS Team and NSIS Community (2020). *Nullsoft Scriptable Installer System (NSIS)*. `https://nsis.sourceforge.io`.

R Core Team (2020a). *foreign: Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...* R package version 0.8-76.

R Core Team (2020b). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rosling, H. (2010). *The Joy of Stats*. [BBC Video]. Available from `https://www.gapminder.org/videos/the-joy-of-stats/`.

The GTK+ Team (2020). *GTK*. `https://www.gtk.org/`.

The jamovi project (2020). `https://www.jamovi.org`.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Verzani, J. (2019). *gWidgets2: Rewrite of gWidgets API for Simplified GUI Construction*. R package version 1.0-8.

Verzani, J. (2020). *gWidgets2RGtk2: Implementation of gWidgets2 for the RGtk2 Package*. R package version 1.0-7.1.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Wickham, H. and Bryan, J. (2019). *readxl: Read Excel Files*. R package version 1.3.1.

Wickham, H. and Grolemund, G. (2017). *R for Data Science*. O'Reilly Media.

Wickham, H., Hester, J., and Francois, R. (2018). *readr: Read Rectangular Text Data*. R package version 1.3.1.

Wills, J. (2020). *NZGrapher*. `https://grapher.jake4maths.com`.

Acknowledgements

iNZight is a free to use, open source software. The work would not have been possible without the support of the University of Auckland, Census at School, ..., Statistics New Zealand, and the Australian Bureau of Statistics.