iNZight User Guides: The Basics

This section will outline the very basics of iNZight. Thanks to the simplicity of the design, you will be exploring your data within minutes!

Table of Contents

Getting started with iNZight

The first thing you'll need to do after having downloaded and installed iNZight will be to launch iNZight. This process varies slightly depending on the operating system you are using.

Windows

When you installed iNZight, the installer should have created desktop icons. Just double-click on **iNZight** to launch the program.

Alternatively, you can start iNZight by going to **Start** > **iNZightVIT** (you might need to go to All Programs first) > **iNZight**.

Mac OS X

iNZight on Mac is located in **Applications** > **iNZightVIT**.

You can start iNZight by double-clicking on **iNZight**.

Unknown Developer Warning

If you're unable to start iNZight due to an "Unknown Developer" warning, simply **Right-click** the iNZight icon, and click **Open**. In the window that pops up, click **Open** to allow iNZight to run.

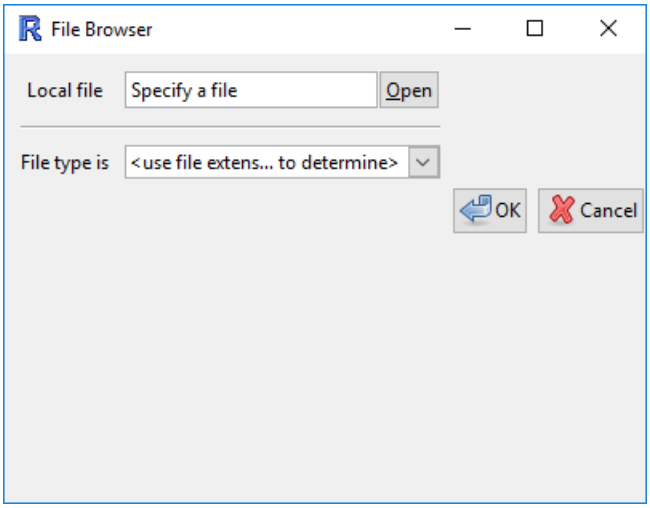
You should only have to do this the first time you start iNZight.

Loading data

The first thing you need to do after starting iNZight is to load a data set to work with. We've made this as easy as possible, and iNZight recognizes several file types, including CSV and Excel (on Windows).

Loading your own data

If you have a data set you would like to load into iNZight, go to the **File** menu and select **Import Data**. This will open the following window:

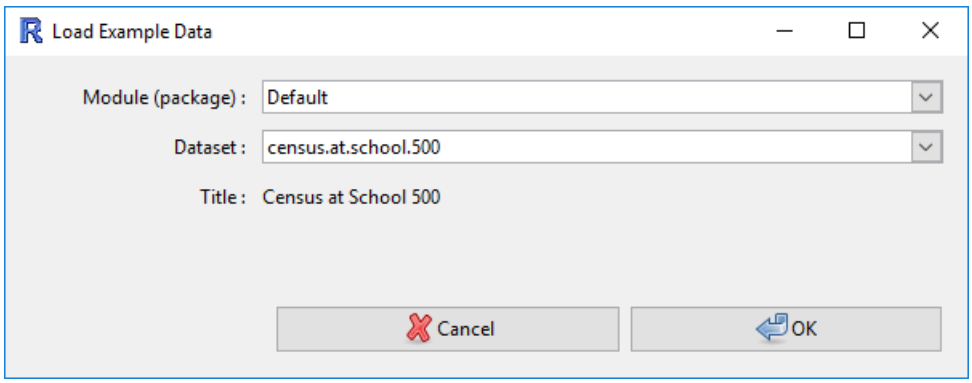


In the window that pops up, click **Open**, and then use the browser to find your data, and click **OK**. Click **OK** again, and your data will appear in iNZight.

Example data sets

To make it even easier to get started with iNZight, we've implemented a Example Datasets tool that allows you to quickly load several data sets that are used in the examples throughout the website.

To load an example data set, go to the **File** menu and click **Example Data**. You'll be presented with this window:



In the example above, we've selected the "Census at School 500" dataset from the Default iNZight module. We will be using this dataset a lot for many of our examples.

Video Demonstration

The following ***video*** demonstrates this process to load the data set *Census at School-500.csv* which you can find in the **Data** folder that came with iNZight.

NOTE: the video was made using an older version of iNZight, so there will be some minor differences in the appearance.

Plotting One Variable

Using iNZight, it is easy to create a graph of your variables. Simply **drag and drop** a variable name into the "Variable 1" slot, and iNZight will automatically draw the appropriate graph depending on the type of variable:

- **Numeric variables**: these will produce a **dot plot**.
- **Categorical variables**: also referred to as **factors**, these will produce a **bar plot**.

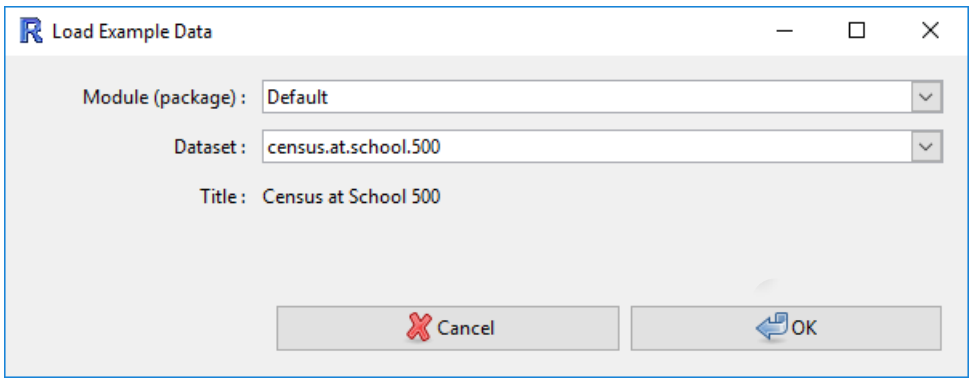
You can also obtain a numeric summary of the variable by clicking the **Get Summary** button at the bottom of the iNZight window. Note that you have to have dragged a variable into Variable 1 first.

The following ***video*** demonstrates how to use iNZight to plot a single variable.

NOTE: the video was made using an older version of iNZight, so there will be some minor differences in the appearance.

You can follow along

1. Load the Census at School 500 data set from **File** > **Example Data**:



2. **Click and drag** the variable *height* to the **Variable 1** slot to produce a dot plot.
3. **Click and drag** the variable *cellsource* to the **Variable 1** slot to produce a bar plot.
4. Click the **Get Summary** button to get a text summary of the graph.

Plotting Two Variables

Plotting two variables is just as easy as plotting one when you use iNZight. Simply drag a second variable name into the **Variable 2** slot, and again iNZight will automatically create the appropriate plot. Depending on the type of variables, you will get either a side-by-side **bar plot**, **dot plot** or **scatter plot**.

If both variables are categorical, then the resulting graph will be a side-by-side **bar plot**.
If both variables are numeric, iNZight will draw a **scatter plot**.
If one variable is numeric and the othewr is categorical, iNZight will draw a **set of dot plots**, one for each group (or category).

You can also obtain numerical summaries for the variables by clicking on the **Get Summary** button at the bottom of the window.

This is all shown in the ***video*** below.

Subsetting Your Data

To find trends in your data, it is often necessary to subset the full data set into levels of a categorical variable. iNZight makes this easy by offering two **subset by** slots. Simply **drag and drop** variable names into them to subset the data set.

The following video explains this in more detail.

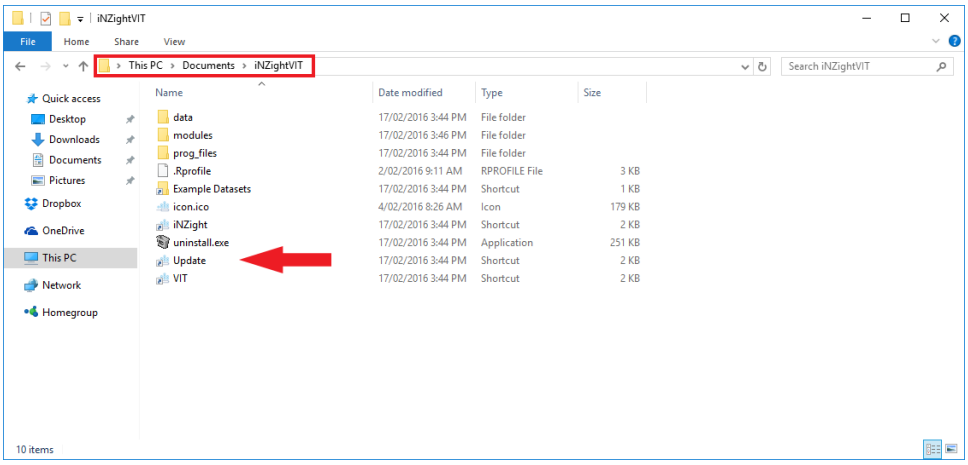
Updating iNZight

We are continually working on iNZight by creating new features and fixing issues here and there. To ensure you stay up-to-date with the latest changes, you can update your copy of iNZight by running the automatic updater.

On Windows

To update iNZight, either:

1. Go to **Start** > **All files** > **iNZightVIT**, and click **Update**, or
2. Assuming you installed iNZight to the default directory, go to **My Documents** > **iNZightVIT** and double-click **Update**. This is shown below:



3. If you can't find the **iNZightVIT** folder in either of those locations, go to the directory where you installed iNZight. If you didn't install iNZight, you might need to ask your system administrator to do it for you.

On Mac

Open **Applications** > **iNZightVIT** and double-click on **Update**.

Unknown Developer Warning

If you're unable to start iNZight due to an "Unknown Developer" warning, simply __Right-click__ the Update icon, and click __Open__. In the window that pops up, click __Open__ to allow iNZight to update.

You should only have to do this the first time you run the updater.

File Menu

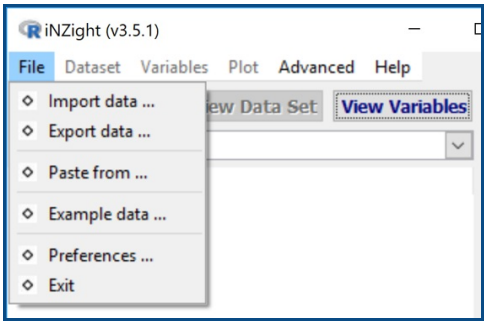
(*Getting data into and out of iNZight (Import, Paste, Examples, Export); Preferences=program controls*)

The **File menu** provides 4 ways of getting data into iNZight

- ***Importing a file on your computer*** (*Import data …*)
- ***Importing a file from an internet url*** (*Import data …*)
- ***Copying and pasting*** data from programs like from Excel or Google sheets (*Paste from …*)
- Built-in ***Example data sets*** (*Example data …*)

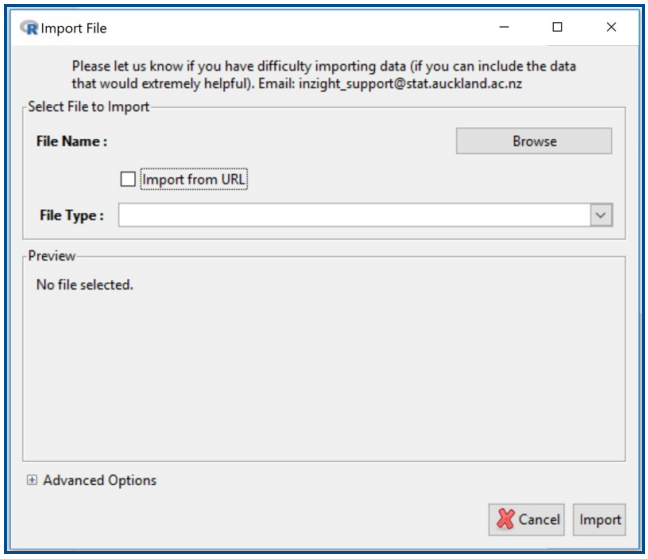
The **File menu also** allows for ***saving/exporting*** data. e.g., if you have modified it in some way in iNZight (*Export data …*)

Contents

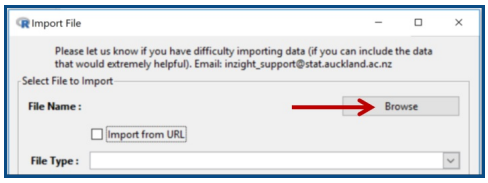


- **Import data ...**(*from a file on computer or internet*)
  - **File types**
  - **Metadata in csv files**
- **Export data ...**(*useful if have modified the dataset*)
- **Paste from ...** (*import by pasting copied data*)
- **Example data ...** (*use an example dataset*)
- **Preferences ...**(*change program behaviour*)
- **Exit ...** (*quit iNZight*)

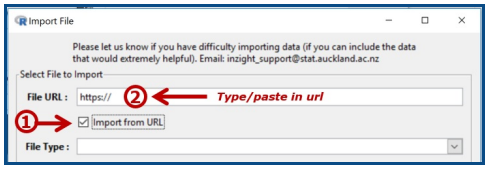
Import data



By default imports ***from a file*** on your computer. Throws up a ***browser*** button to initiate a dialog for navigating to the file you want.



On clicking the **Import from url** *checkbox* the behaviour changes (as below)



Type or paste in the url address of the file you want to import
(*when pasting a full url, ensure the prepopulated "https:// " is overwritten*)

File types

NZight assumes that data sets are in **rows = cases** by **columns = variables** format

iNZight can import files of the following types:

- Comma Separated Values text files (.csv)
- Tab-delimited Text files (.txt)
- 97-2003 Excel files (.xls)
- 2007 Excel files (.xlsx)
- SPSS files (.sav)
- SAS Data files (.sas7bdat)
- SAS Xport files (.xpt)
- STATA fles (.dta)
- R Object files (.rds)
- RData files (.RData, .rda)

By default, iNZight identifies the type of the file being imported by looking for one of these filename extensions. iNZight attempts to read the file and populate the **Preview panel**. If no preview appears iNZight has been unable to read the file.

Click the **Import** button (at bottom right) when you are happy with what appears in the Preview panel.

The **Advanced Options** at the bottom-left of the Import dialog box (Click on the " **+**" to expand) allows you to change default behaviour.

Metadata in csv files

(**Metadata** is *information about the data.*)

When .csv files and are .txt files are imported into iNZight, all lines in the data file beginning with a "#" are ignored (there is one exception below).

This enables you to include **metadata** in your .csv or .txt data files by starting each line of metadata information with a "#".

With .csv files, you can go further and include ***instructions telling iNZight to treat variables in various ways.*** This is particularly useful for instructing iNZight to order the levels of a categorical variable in a desired way, rather than the default way.

Lines starting with #' @ are interpreted by iNZight's smart read function as containing instructions of this form

Example:
We will use the following lines from the top of the file at ... **https://www.stat.auckland.ac.nz/~wild/data/test/Census%20at%20School-500&meta.csv** to show how this works.

```
## --- Meta Data --- ##
#' @factor gender[male,female]
#' @factor cellsource[pocket,parent,job,other]
#' @factor getlunch[home,tuckshop,dairy,school,friend,none]
#' @numeric age
#' @factor year
#' @factor school=year[primary=5|6,intermediate=7|8,high=9|10|11]
#' @factor travel[walk,bike,car=motor,public=bus|train,other]
#' @factor cellsource
#' @numeric phonebill=cellcost
## --- Data --- ##
```

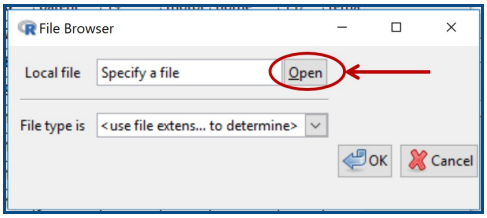
Note: *By default in R, and therefore iNZight, the categories (levels) of a categorical variable (factor) are ordered alphanumerically.*

The 1st (real) line above is an instruction for **gender** to be treated as a categorical variable (factor) with levels in the order "male, female", rather than the alphabetic default "female, male". The next 2 lines are doing the same sort of reorderings for **cellsource** and **getlunch**.

year in the data file has numeric values ranging from 4 to 13 and so by default will be treated as a numeric variable. "#' @factor year " is an instruction to treat it as a categorical variable (factor).

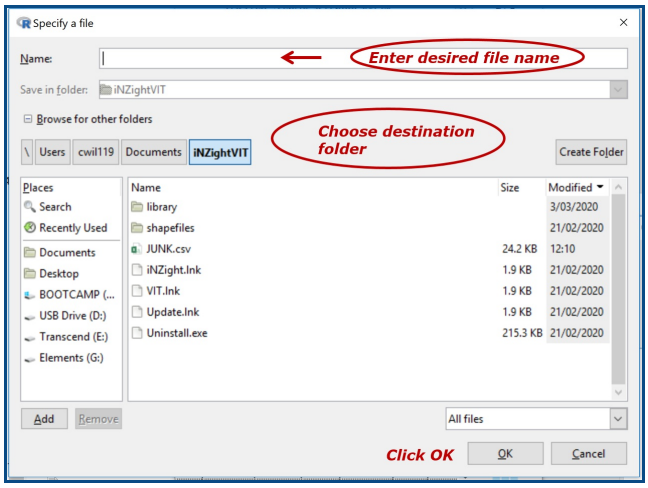
The next line, "#' @factor school=year[primary=5|6,intermediate=7|8,high=9|10|11]" is an instruction to construct a new categorical variable called **school** from **year** in which students in year-levels 5 & 6 become a category called "primary", ..., those in year-levels 9 to 11 become a category called "high". Year-levels 4, 12 an 13 are not mentioned in the statement so those categories remain unchanged and sort after those mentioned in the statement.

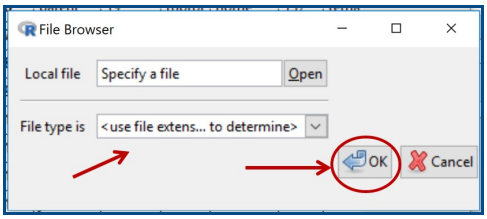
The next line redefines **travel** by combining categories "bus" and "train" into a category called "public".



Export data ...

Clicking "Open" initiates this dialog ...





This brings you back to ...

Choose from allowable file extensions and **click OK**

(*If you have already put the correct extension on your filename you can click OK without choosing an file type extension*)

Paste from ...

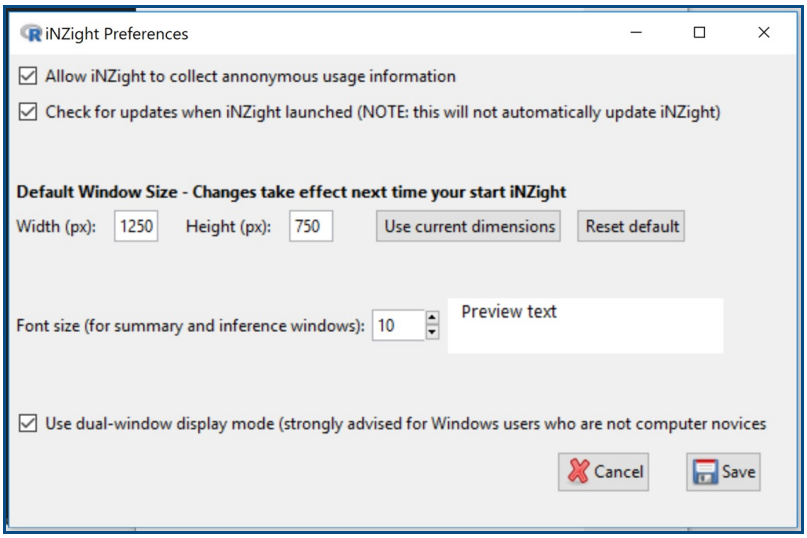
This enables copying and pasting data from sources like Excel files or Google sheets

Click **Load** when finished

Example data ...

Enables the loading of data files stored inside iNZight.

First select a Module (package) and *then* a **Dataset** stored with that Module.



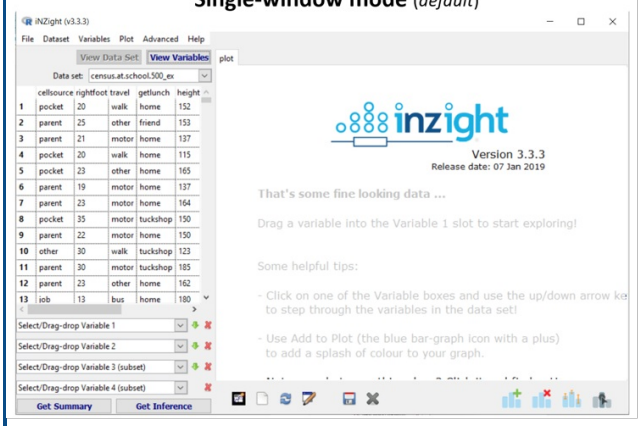
Preferences ...

Preferences enables changing some of the behaviours of iNZight such as default window and font sizes.

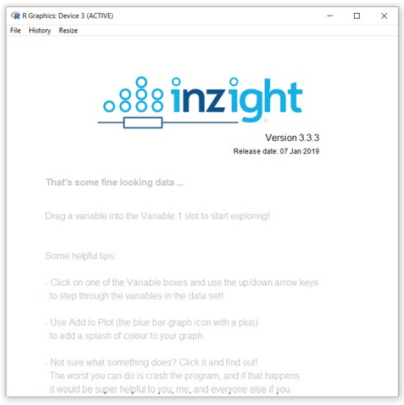
Preference changes do not take effect until the next time you start up iNZIght.

Single window versus Dual-window display
While *single-window* is the default, *dual-window* mode is better for all Windows users except computer novices.

Single-window mode (*default*)



Dual-window mode (*better for Windows machines*)

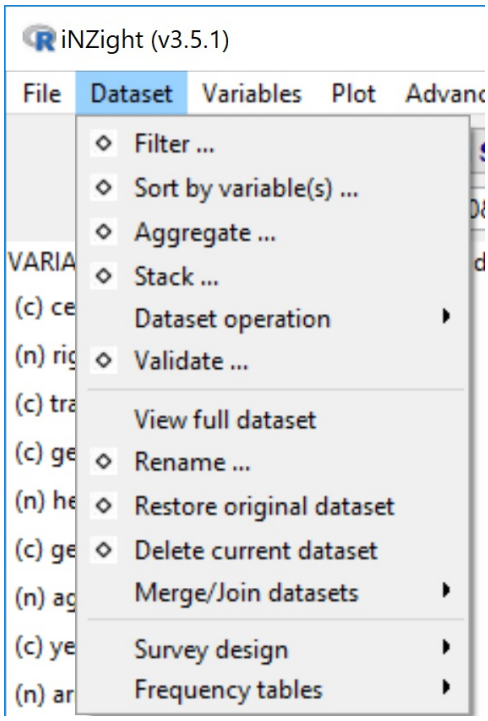


(*separate floating windows*)

Dataset Menu

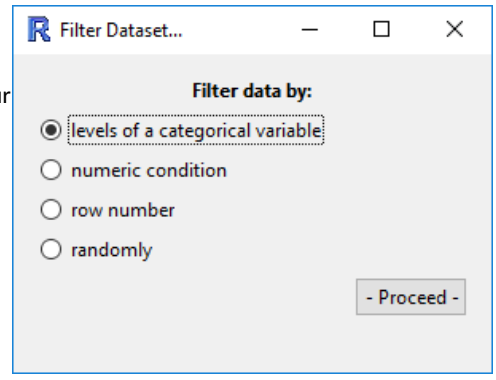
(*Organise and restructure your data, specify special structures, and filter out unwanted observations*)

Contents



- **Filter ...** (*dataset*)
  - **Levels of a categorical variable**
  - **Numeric condition**
  - **Row number**
  - **Randomly**
- **Sort (*dataset*) by variable(s) ...**
- **Aggregate ...** (*data*)
- **Stack ...** (*variables*)
- **Dataset operations:**
  - **Reshape dataset**
    - **Wide to long**
    - **Long to wide**
  - **Separate column (into) ...**
    - **into several columns**
    - **into several rows**
  - **Unite columns ...**
- **Validate ...** (*dataset*)
- **View full dataset**
- **Rename ...** (*dataset*)
- **Restore original dataset**
- **Delete current dataset**
- **Merge/Join datasets:**
  - **Join datasets** (*several **types** of join*)
  - **Append rows**
- **Survey design** (*for data from complex survey designs*)
  - **About survey data**
  - **Specify design ...**
  - **Specify replicate design ...**
  - **Post stratify ... (*or rake or calibrate*)**
  - **Remove design ...**
- **Frequency tables** (*values plus frequencies data format*)
  - Expand table
  - Specify frequency column
  - Remove frequency column

Filter Dataset

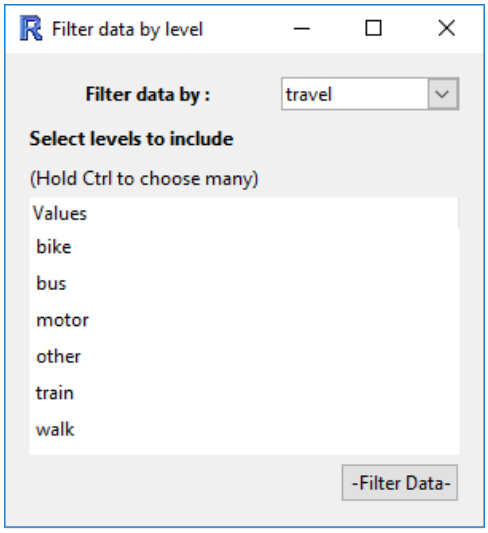


This tool provides several methods for filtering the dataset. The window that opens has four options for you to choose from:

- **Levels of a categorical variable**
- **Numeric condition**
- **Row number**
- **Randomly**

(Filter by) Levels of a categorical variable

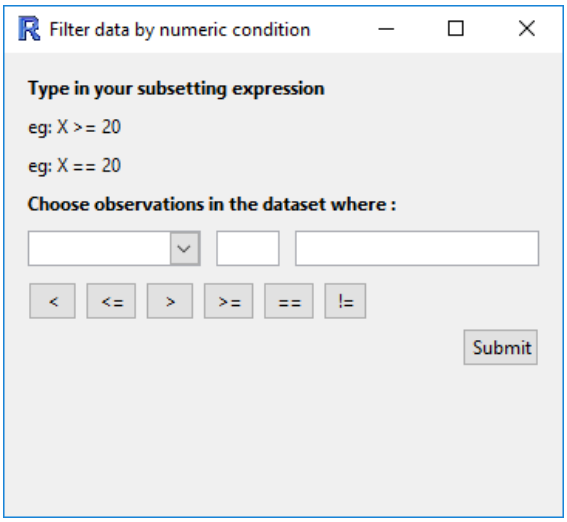
After selecting a categorical variable from the drop down box, you can select which levels you want ***to keep*** in the data set.



(Filter by) Numeric condition

This allows you to define a condition with which to filter your data. For example, you could include only the observations of `height` over 180 cm by

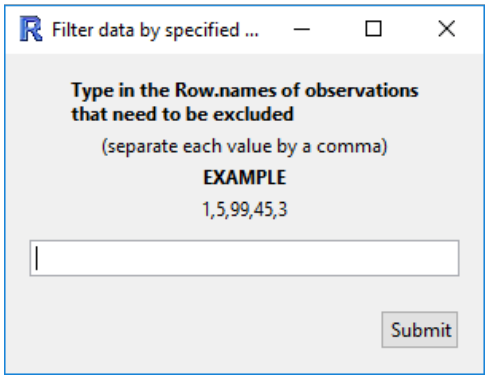
- selecting `height` from the drop down menu,
- clicking on the `>` symbol, and
- entering the value `180` in the third box.



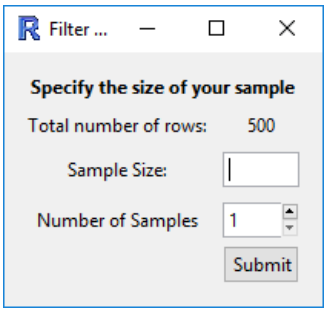
(Filter by) Row number

Exclude a range of row numbers as follows:

- Entering 101:1000 (and then **Submit**) will exclude all rows from 101 to 1000
- Similarly, 1, 5, 99, 101:1000 will exclude rows 1, 5, 99, and everything from 101 to 1000



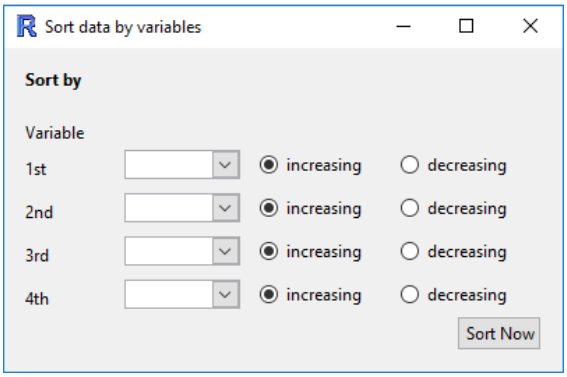
(Filter by) Randomly



Essentially, this allows you to perform bootstrap randomisation manually. The current behaviour is this:

- "Sample Size", *n*, is the number of observations to draw for each sample,
- "Number of Samples", *m*, is the number of samples to create in the new data set.
- The output will be a data set with *n* x *m* rows, which *must be smaller than the total number of rows in the data set*.
- The observations are drawn randomly *without replacement* from the data set.

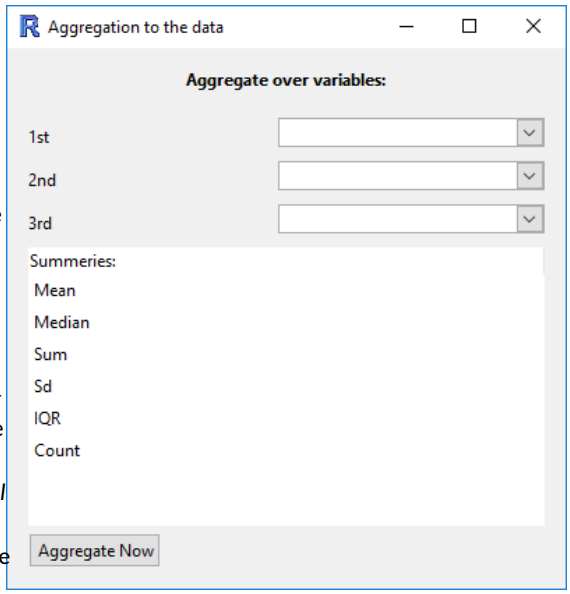
Sort data by variables



Sort the rows of the data by one or more variables

- The ordering will be nested, so that the data is first ordered by "Variable 1", and then "Variable 2", etc.
- For categorical variables, the ordering will be based on the order of the variable (by default, this will be alphabetical unless manually changed in "Manipulate Variables" > "Categorical Variables" > "Reorder Levels").

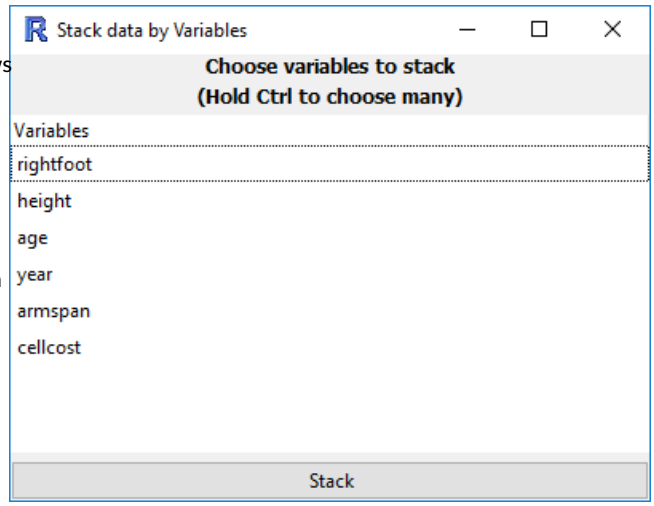
Aggregate data



This function essentially allows you to obtain "summaries" of all of the numeric variables in the data set for combinations of categorical variables.

- *Variables*: if only one variable is specified, the new data set will have one row for each level of the variable. If two (or more) are specified, then there will be one row for each combination. For example, the categorical variables `gender = {male, female}` and `ethnicity = {white, black, asian, other}` will result in a data set with 2x4 rows.
- *Summaries*: each row will have the chosen summaries given for each numeric variable in the data set. For example, if the data set has the variables `gender (cat)` and `height (num)`, and if the user selects `Mean` and `Sd`, then the new data set will have the columns `gender`, `height.Mean` and `height.Sd`. In the rows, the values will be *for that combination of categorical variables*; the row for `gender = female` will have the mean height *of the females*, and the standard deviation of height *for the females*. A visual example of this would be do drag `height` into the Variable 1 slot, and `gender` into the Variable 2 slot. Clicking on "Get Summary" would provide the same information. The advantage of using Aggregate is that the summaries are calculated *for every numeric variable in the data set*, not just one of them.

Stack variables



Convert from *table form* (rows corresponding to subjects) to *long form* (rows corresponding to observations).

In many cases, the data may be in tabular form, in which multiple observations are made but placed in different *columns*. An example of this may be a study of blood pressure on patients using several medications. The columns of this data set may be: `patient.id`, `gender`, `drug`, `Week1`, `Week2`, `Week3`. Here, each patient has their own *row* in the data set, but each row contains three observations of blood pressure.

patient.id	gender	drug	Week1	Week2	Week 3
1	male	A	130	125	120
2	male	B	140	130	110
3	female	A	120	119	116

We may want to convert to *long form*, where we have each observation in a new row, and use a categorical variable to differentiate the weeks. In this case, we would select `Week1`, `Week2`, and `Week3` as the variables in the list. The new data set will have the columns `patient.id`, `gender`, `drug`, `Stack.variable` ("Week"), and `stack.value` ("blood pressure").

patient.id	gender	drug	stack.variable	stack.value
1	male	A	Week1	130
1	male	A	Week2	125
1	male	A	Week3	120
2	male	B	Week1	140
2	male	B	Week2	130
2	male	B	Week3	110
3	female	A	Week1	120
3	female	A	Week2	119
3	female	A	Week3	116

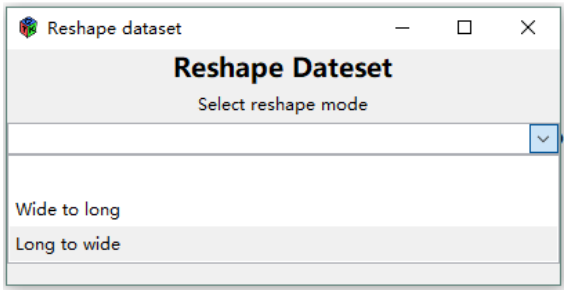
Of course, you can rename the variables as appropriate using "Manipulate Variables" > "Rename Variables".

Dataset operations

It offers three types of tools for the users to modify their dataset:

- **Reshape dataset**
- **Separate column**
- **Unite columns**

Reshape dataset

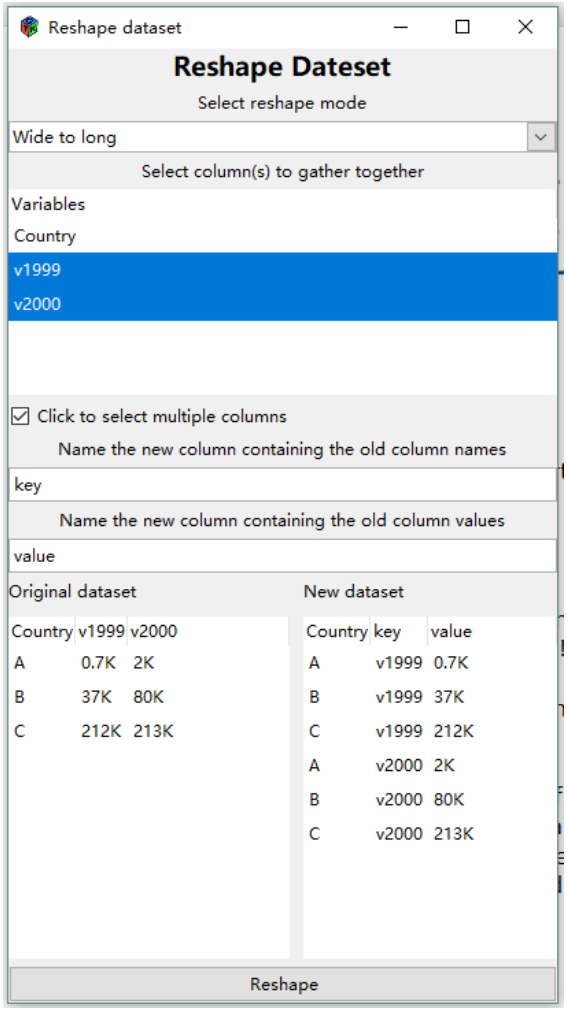


- **Wide to long**
- **Long to wide**

(Reshape) Wide to long

This allows you to select a column or multiple columns from your dataset.

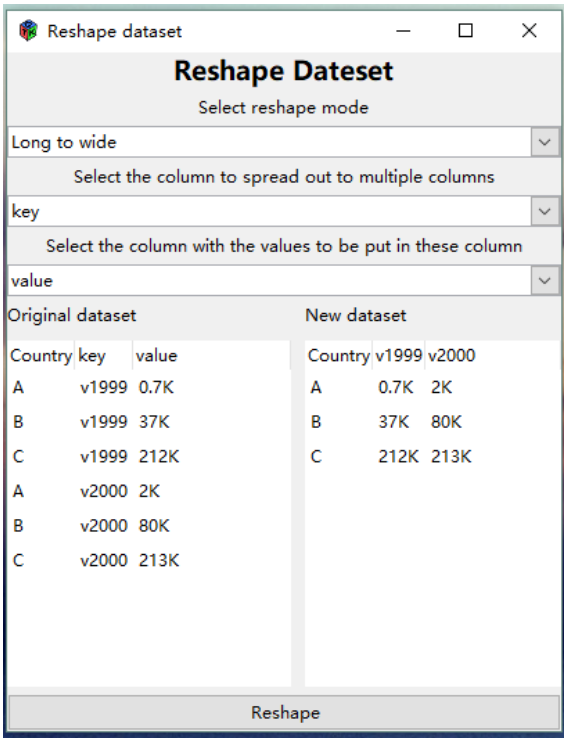
- One new column (default name `key`) is populated by the column name(s) of the selected column(s)
- The other new column (default name `value`) will contain the column value from the selected columns.
- The selected column(s) will be removed and two new columns will be added to the dataset.
- A preview panel shows what the new dataset will look like.



(Reshape) Long to wide

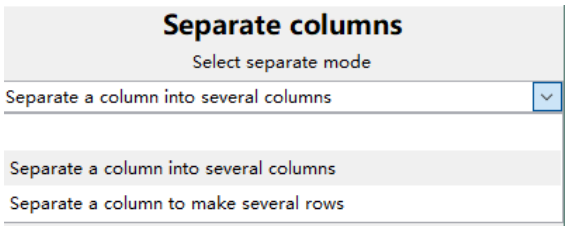
You can select a column to spread out into multiple columns (the column is named `key` in the example). It will use the column values of the selected column as a set of names for new columns.

You then select another column with corresponding values to be put into the new columns (the column is named `value` in the example).



Separate column into ...

- **Separate a column into several columns**
- **Separate a _column into several rows**



Separate a column into several columns

Allows you to separate a column into several columns using a user-defined separator.

- It will separate at every instance of the separator until no further separators are found

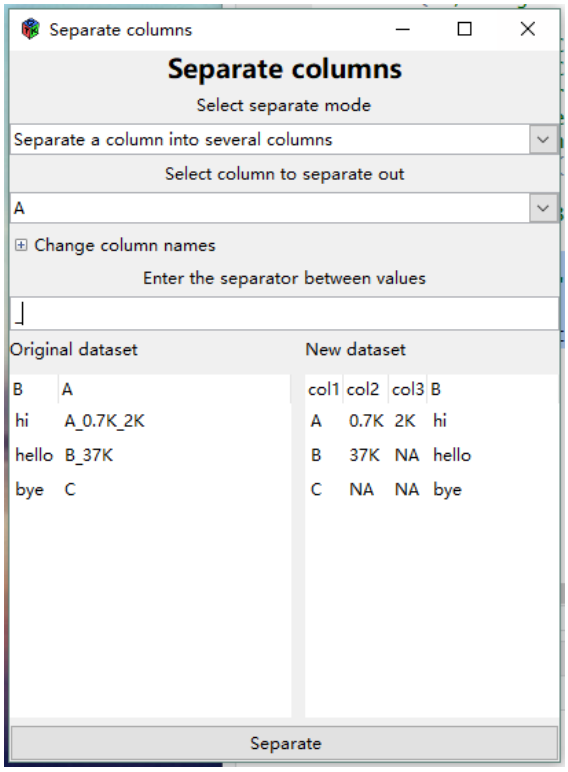
If no separator is found, the additional columns formed will contain `NA` s.

In the example on to the right, we have asked to separate column **A** using an underscore ("_") as a separator.

Because only column **A** is being separated, column **B** (or any other columns) is left unchanged in the resulting new dataset

The maximum number of fields in column **A** after separation is 3 ("A_0.7K_2K") so column **A** in the original dataset is being replaced by 3 columns with default column names (Col1, etc).

- Expanding **Change column names** (click the "+") allows you to change the default column names to something else.

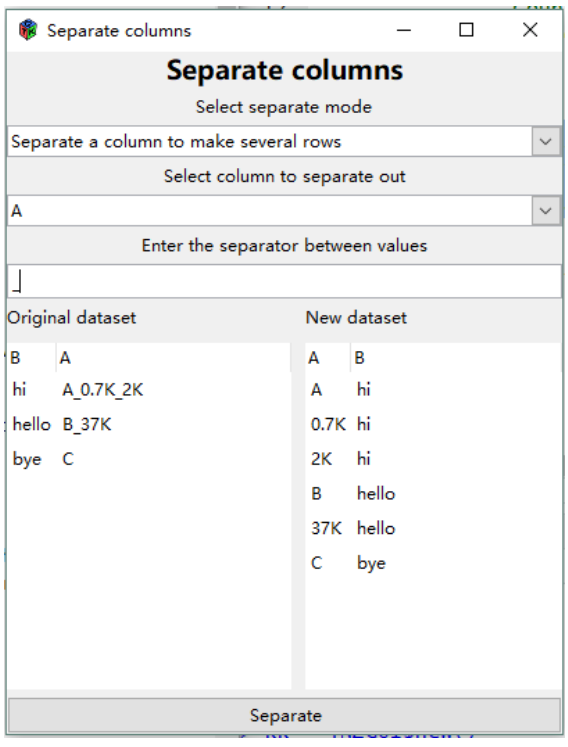


Separate a column into several rows

Instead of forming more columns this version of Separate keeps the same number of columns, with the same names, but writes more rows.

Using the same data as in the example above, the entry "A_0.7K_2K" in column **A** in the original dataset results in 3 rows in column **A** in the new dataset.

- The corresponding entries in any other columns (e.g. column **B** in the example) ***are duplicated***.

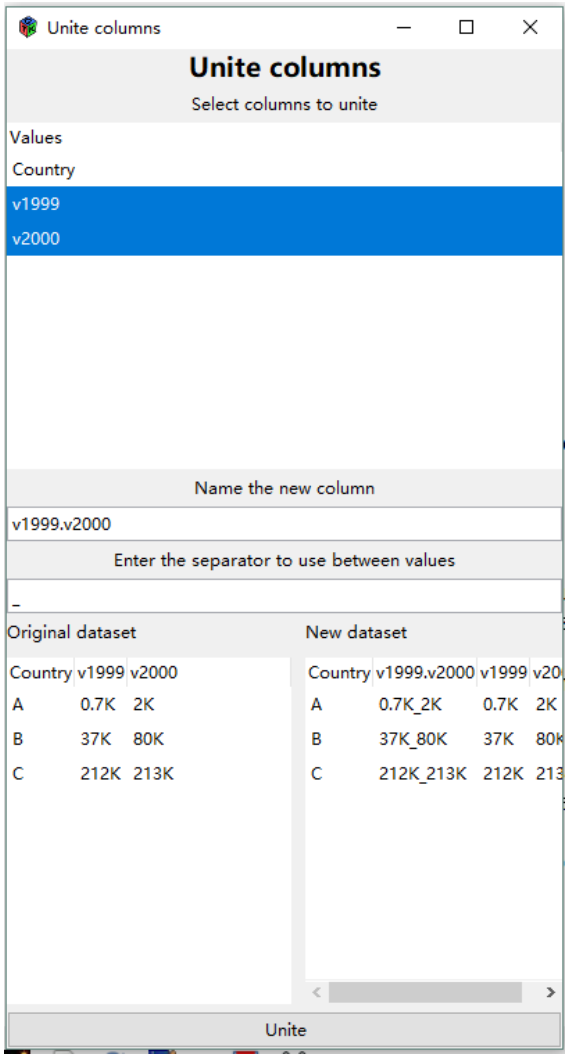


Unite columns

Allows you to select multiple columns and "unite" them using a defined separator (defaults to "_"). The united column name will be the combination of the selected columns with a "." in between.

In this example, column "v1999" and column "v2000" are united by ". The new column name is "v1999.v2000"._

- It is allowable to have ***no separator***. Just clear the ***Enter the separator*** field (delete the "_")



Merge/Join datasets

- **Join datasets**
- **Append rows**

Join Datasets

This "joins", or brings together, information in two data sets: the current dataset in iNZight and a newly imported dataset (read in using the **Import data** facility) shown at the mid-right.

Left Join: The most important joining method is called a ***Left Join***, the main purpose of which is to add new variables to the original dataset by extracting the information from the new dataset.

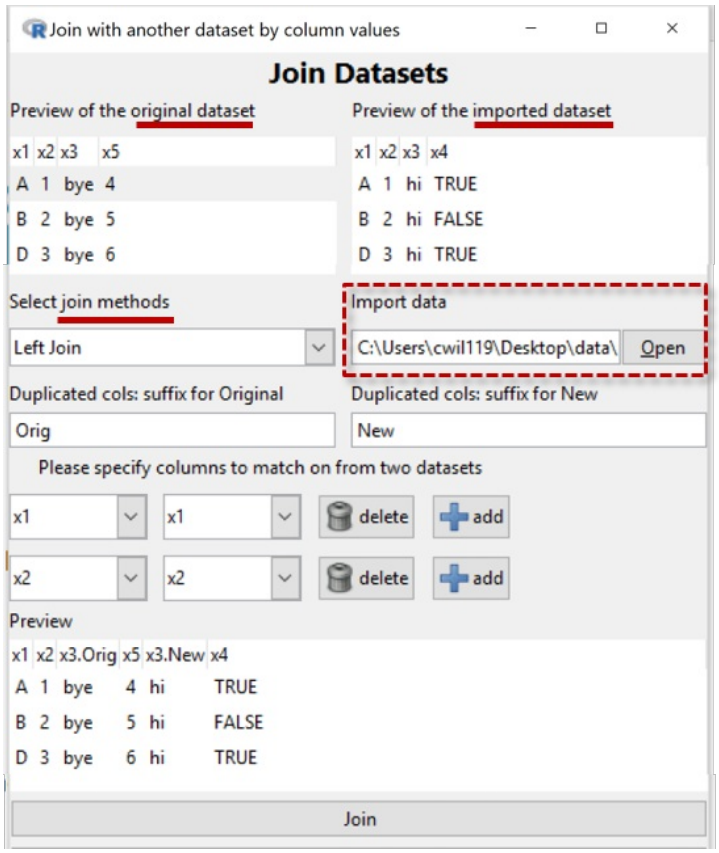
Matching rows: (***Not yet fully implemented in Lite.***) The main problem is to identify what pieces of information belong together. The most straightforward case occurs where there is a variable in the original dataset which is a unique identifier. If that variable is also in the imported dataset (even if under a different name) we can use it to match up the data which belongs to the same unit/entity.

To partially automate the process, iNZight looks for variables with the same name in both datasets (*originally x1, x2, and x3 in the Example to the right*) and offers those for determining matches.

In the Example, we have rejected x3 using the delete button beside it and so have effectively told the program that it is units with the same values of both x1 and x2 that belong together.

- The **Preview** panel shows us the effects of our choices

Click the **Join** button at the bottom once you are happy with the way the data is being joined.



The details of how the data is treated depend on the type of Join and we will **document that** after finishing describing the Example.

In the Example, x4 is a new variable so that has been added to the preview-dataset. A complication is x3 which is in both datasets but with different values for the "same" units. So the program has decided to make two variables, one for the x3 values from the original dataset and one for the x3 values from the new dataset.

Types of Join

Left Join

- The joined dataset has rows corresponding to ***all of the rows in the original dataset*** and all of its columns.
- Rows of the ***new*** dataset ***that do not have a match*** in the original dataset ***are not used***.
- The joined dataset ***also has the columns from the new dataset*** that were not used for matching.
- Rows in the original that ***have no match*** in the newly imported dataset get `NA` s for the additional columns
- **Warning:** Rows from the original dataset that have ***more than one match*** in the new dataset generate multiple rows in the joined dataset (which invalidates many simple analyses). For example, if there are 3 matches then the original (single) row will be replaced by 3 rows. The cell-values for the additional columns will be obtained from the new data set and the values for the original columns from repeating the original cell values.

How other joins differ from the Left Join

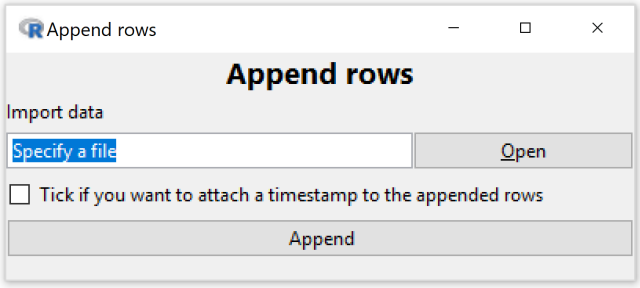
- **Inner Join:** Only use rows corresponding to matches between the two datasets
- **Full Join (Outer Join):** Also use all the non-matching rows from both data sets

[**Right Join:** iNZight does not have this. Just import the datasets in the reverse order and use a left join.]

The following are just used to filter data. No columns are added to the join from the new dataset.

- **Semi Join:** Use only rows in the original which have ***a match*** in the new
- **Anti Join:** Use only rows in the original which have ***no match*** in the new

Append rows



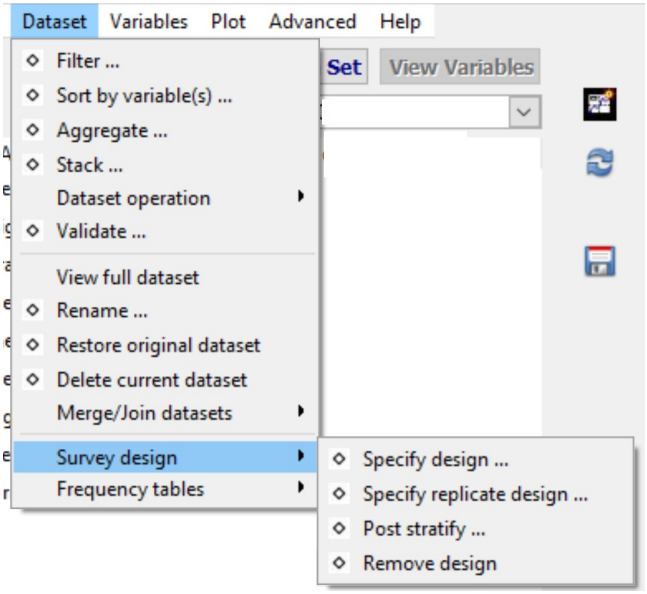
Adds to the bottom of the original dataset rows from a newly imported dataset (imported using the Import facility provided).

- The column names from both datasets are matched so the correct data goes into the correct columns
- Columns that appear in one dataset and not the other *will* appear in the result
- An error will be reported and not appending will be possible if two column names match but their types (e.g. numeric or categorical) do not match
- If the Tick-box option is selected a timestamp variable called **When_Added** will be added to the dataset recording when the new observations were added
- If there is already a variable called **When_Added** present in the original data (must be of type date-time, "(t)" in View Variables) the new timestamps will be appended to that existing variable

Where data is periodically being added to a dataset, this facility can be used to keep track of when each row was added -- thus facilitating analyses of the data as it was up until any particular time point

Survey design

- **About survey data**
- **Specify design ...**
- **Specify replicate design ...**
- **Post stratify ... (*or rake or calibrate*)**
- **Remove design ...**



About survey data

It is important that specialist methods be applied when analyzing data obtained using complex survey designs. Failure to do so typically leads biased estimates and incorrect standard errors, confidence intervals and p-values.

When a survey design has been defined almost all relevant parts of iNZight will ***apply analysis and graphics methods appropriate*** for data obtained using ***this survey design*** by applying functions in R's **survey package**.

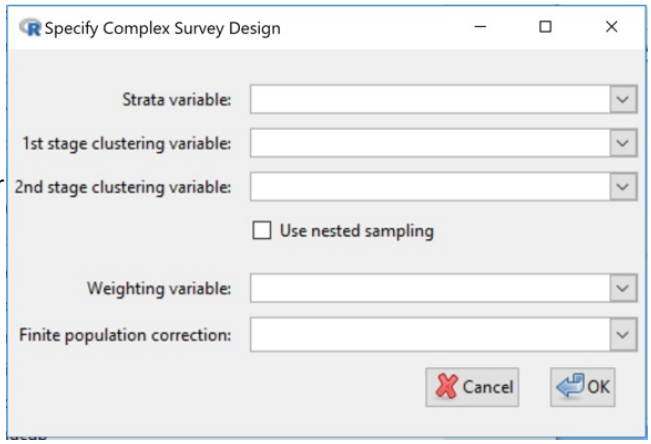
Important difference for **Get Summary**. Whereas, generally, iNZight's **Get Summary** provides summary information about the dataset itself, when a survey design is specified **Get Summary** provides estimates of population quantities -- clearly labeled as such. (Raw summaries of survey data are often meaningless because of unequal probabilities of selection.)

Regular statistical software analyses data as if the data were collected using simple random sampling. Many surveys, however, are conducted using more complicated sampling methods. Not only is it often nearly impossible to implement simple random sampling, more complex methods are more efficient both financially and statistically. These methods use some or all of *stratified sampling*, *cluster sampling* and *unequal sampling rates* whereby some parts of a population are sampled more heavily (i.e. with *higher probabilities of selection*) than others parts. These sampling features have to be allowed for in the analysis. While sometimes it may be possible to get reasonably accurate results using non-survey software, there is no practical way to know beforehand how far wrong the results from non-survey software will be.

- See **Brief introduction to survey analysis ideas**
- For an in-depth account, see **"*iNZight and Sample Surveys*"** by **Richard Arnold**.

Survey designs are typically specified to analysis programs either by **specifying a survey design** in terms of weighting, strata and clustering variables etc in the data set, or by selecting a set of variables in the dataset containing so-called **replicate weights**. **Post-stratification/raking/calibration** facilitate using additional population information (external to the survey data) to improve survey estimates. This is done after a survey design has been specified (by either method).

Specify design



Specifying a survey design in terms of weighting, strata and clustering variables etc.

iNZight's survey methods cater for simple random sampling, stratified random sampling (random sampling within each population stratum), cluster sampling and multistage cluster sampling, and complex designs involving cluster sampling within population strata.

- **Strata variable:** If **stratified sampling** has been performed, use to select the variable that specifies which stratum each observation comes from. (This variable can be either numeric or categorical.)
- **1st stage clustering variable:** If **cluster sampling** has been performed, use to select the 1st-stage clustering variable; this specifies which **1st-stage cluster** each observation comes from. (Clustering variables can also be either numeric or categorical.)
- **2nd stage clustering variable:** If two or more stages of cluster sampling have been performed, use to select the 2nd-stage clustering variable (specifies which **2nd-stage cluster** each observation comes from). Any further levels of cluster sampling (3rd stage, etc., are not used)
- **Use nested sampling:** Quite often, compilers of survey data "reuse" cluster names from stratum to stratum. Let us take, as an example, a survey in which American states for the starts and counties form the clusters. Sampled counties from Washington State may be given a County value of 1, 2, ... and counties from Arizona may also be given a County value of 1, 2, ... Clearly County 1 from Washington refers to an entirely different county from County 1 from Arizona even though they have the same value of the County variable. -- ***Click*** the **Use nested sampling** check-box if cluster labels are being recycled/reused in the data.
- **Weighting variable:** If the sampling design used unequal probabilities of selection, use this to select a variable containing the ***sampling weight*** (1 over the probability of selection) for each observation. Certain estimates will be wrong if the sampling weights do not add up to the population size, in particular estimated *population* or *subpopulation* ***totals*** and estimated *population* or *sub****population sizes***.

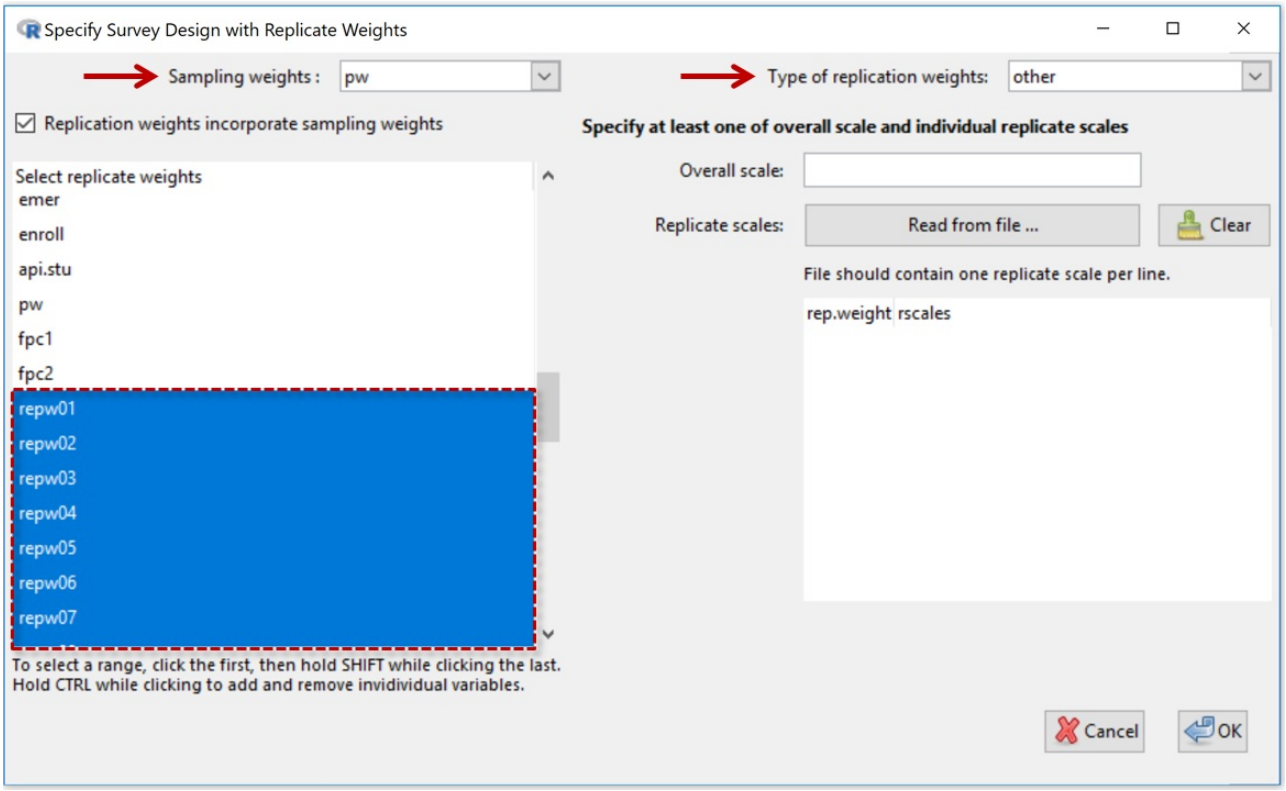
Sampling weights are often *adjusted* to allow for *unit non-response*.

- **Finite population correction (fpc):** Use this when descriptive inferences are wanted about properties of the finite population being sampled and the sample size is an appreciable proportion of the population size (e.g. > 5 or 10%). If stratified sampling has not been used, this variable should contain an estimate of the size of the population being sampled, repeated for every observation. If stratified sampling has been used, the values of this variable should contain an estimate of the size of the population stratum being sampled (differing across strata but constant within each stratum). As an alternative to using population/stratum sizes, proportions *of the total population* being sampled can be used.
  If *one stage cluster sampling has been used*(where we take a random sample of clusters, and a census within clusters), the finite-population-correction selection-box should contain an estimate of the total number of clusters (the same on every row, or the same on every row within a stratum).
  In two stage clustering we subsample within each selected cluster, and so we need to specify two variables: one for the number of clusters (just as in one stage clustering), and a second for the total number of units available for selection within each cluster (the same value on every row within the cluster). If the number of clusters is stored in a variable called **fpc1**, and the number of units within a cluster is stored in a variable called **fpc2**, then type/paste **fpc1+fpc2** in the finite population correction field.

For more information on quantities referred to in the dialog box see the documentation of the **svydesign** function in R's **survey** package.

Specify replicate design

Because making public factors like cluster membership can make it easier than survey agencies are comfortable with to identify individuals, many agencies do not distribute such information to outsiders. Instead they distribute sets of so-called *replicate weights*, slightly varying copies of the sampling weights variable that still enable survey analysis programs to make the proper adjustments to analyses of survey data.



- **Sampling Weights:** See variable selection box at the top-left
- **Replication weights incorporate sampling weights** (*checkbox*)**:** This should be ***checked if*** the replicate weights already include the sampling weights (which is usually the case). ***Uncheck this*** if the replicate weights are very different in size to the sampling weights.
- **Select replicate weights:** The large (lower) panel on the left-hand side displays the names of the variables in the dataset. Use to select the replicate-weights variables. In the example shown the replicate-weights variables were called *repw01*, *repw02*, *repw03*, ...

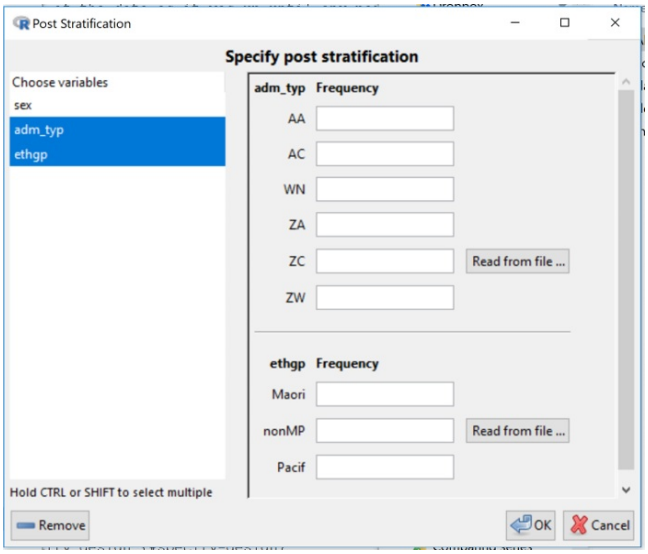
Right-hand panel

- **Type of replication weights:** Depends on the type of replicate weights the person who compiled the dataset has used. Select from list - *BRR*, *Fay*, *JK1*, *JKn*, *bootstrap*, *other*.
- **Overall scales:** Only used for Types *bootstrap* and *other*.

For more information on quantities referred to above

- see the documentation for the **svrepdesign** function in R's survey package and also **here**
- see also the **section on replicate weights** in "***iNZight and Sample Surveys***" by **Richard Arnold**..

Post stratify



This allows for ***poststratifying/raking/calibrating*** a design that has already been defined using either of the methods above above. (Technically the design is updated using the **calibrate** function in R's survey package.)

This allows a data-analyst to improve estimation by augmenting the information in the survey data by adding information on the whole population where this is available from other sources. Categorical variables in the data set are offered as possible poststratification/raking/calibration candidates. Corresponding population counts can be input by typing or reading from files.

In the example in the screenshot, there are 3 categorical variables in the data set offered as possible candidates and 2 have been selected. The design would then be calibrated using user-supplied information on population counts for the all the categories of both **admin_type** and **ethgp**.

This is fine when you only want to use information about single variables, but what if you have population information on the cross-classification (all possible combinations) of **admin_type** and **ethgp**, say? Then you would have to create a new variable in the dataset, called say **admin_type.ethgp** that has all these combinations. This can be done with **Variables > Categorical Variables > Combine categorical variables**.

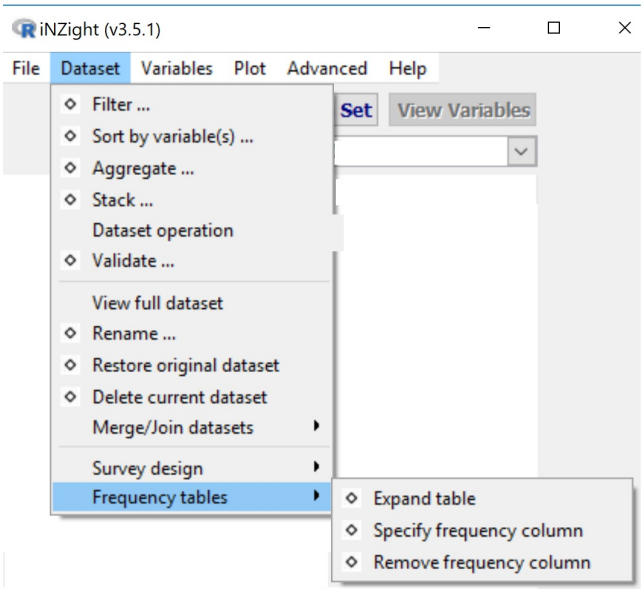
***Warning*:** Currently the new variables have to be set up before specifying a survey design. If you only think about it later you will need to use **Remove design** (next item), set up new variables and then re-specify the design.

For more information on quantities referred to in the dialog box **see here ....**

Remove design

Discard design information and ***revert to using standard methods of analysis***.

Frequency tables (*values plus frequencies data format*)



When a **frequency column** is specified, graphics and summaries respond to a combinations-of-variables-plus-their-frequencies data structure. Only categorical variables are retained.

The **expand table** option can be used to work with both numeric and categorical variables. The storage and efficiency savings of working with a unique combinations of variables plus frequencies will typically be very much smaller here.

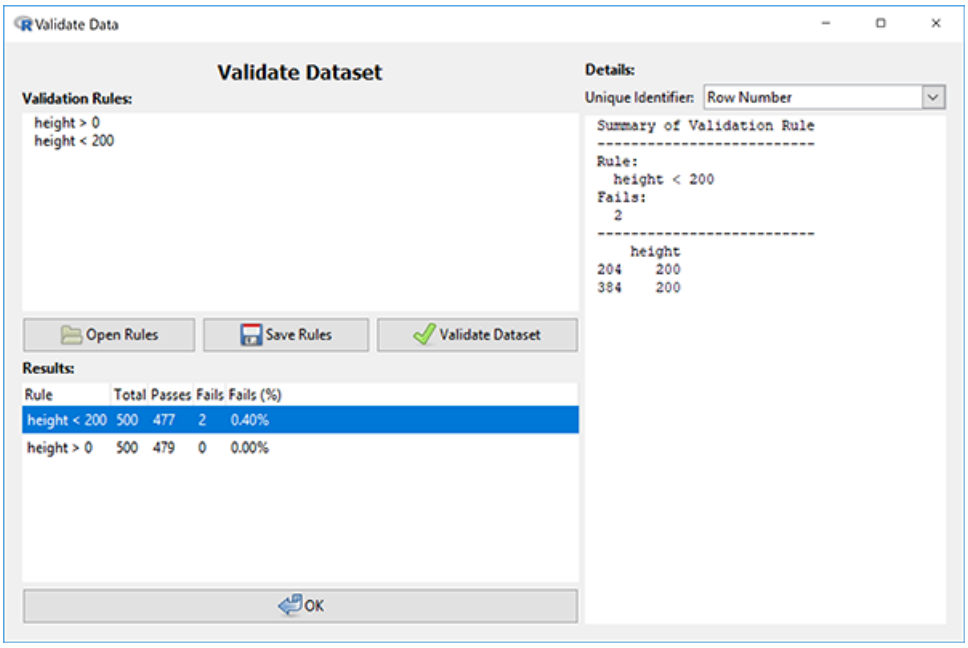
Validate Dataset

Often, we want to validate that the data in a dataset adheres to our expectations about how that data should behave based on external real world-knowledge.

- ***Validating a data set necessarily requires user-supplied validation rules.***

For example, we expect the heights of humans are not negative or someone cannot work for more than 24 hours in a single day.

The validation window gives us the ability to define or import rules, and check whether the data conforms to those rules identifying any exceptions.



- In the validation window, rules can be ***typed*** into the **Validation Rules** text box in the top left ***or imported*** from a text file using the **Open Rules** button.

Using the first example, the rule to check whether heights are above 0 can be written in this textbox as `height > 0` in this text box (given there is a variable named `height` in our dataset).

- **To check all of the rules**that have been defined in this textbox, ***click*** the **Validate Dataset** button.

The results of each rule are presented in a table at the bottom of the window and show the number of observations that were checked ("Total"), the number of passes and failures ("Passes" and "Fails" respectively), and the fail percentage ("Fails (%)"). Initially this table is sorted by failure percentage, but you can click on other column headers to order the list in other ways.

- The **Unique Identifier** selection box at the top right of the validation window allows you to select the name of a variable that contains unique identifiers for the units/cases/rows in the data set.

This is more useful than employing row numbers (the default setting) because unique-identifier values remain unchanged in the data when rows are deleted whereas row numbers will change.

- **Double clicking on a row*of the results table*** will generate a ***detailed breakdown*** of the results in the **Details** section on the right-hand side of the window.

This breakdown will provide details about the observations which failed on that particular rule, giving the row numbers (or unique-identifier values if a unique-identifier has been selected) of these observations and the values used to assess the rule.

Validation rules or changes to imported rule files are ***discarded*** once the validation window is closed.

- If you would like to store the set of rules you have defined or save any changes to an imported rule file, this can be done using the **Save Rules** button.

They are saved into a text file on your computer that can be imported again using the "Open Rules" button or viewed using a text editor.

The rules you use to validate the dataset do not need to be simple comparisons between a variable and a static value as in the previous example. ***More complex rules*** can be built by performing calculations on the variables, e.g. `weight / height^2 < 50` will verify that each observation's body mass index is below 50. The values of each variable contained in the calculation as well as the end result are provided in the detailed breakdown.

- Instead of comparisons between a variable/calculation and a static value, we can ***compare against*** another variable or ***a calculation based on the data***.

For example, to check that the income of an individual (contained in variable `Income`) is no more than 1000 times their number of hours per week (contained in variable `Hours`), we can use the following rule: `Income <= Hours * 1000`. This will calculate a different value to compare income against for each observation.

For more information on what is possible using validation rules, the vignettes and help files of the underlying R package (validate) might be useful: **Introduction to Validate vignette** and **Validate package (in particular, the syntax section of the reference manual)**

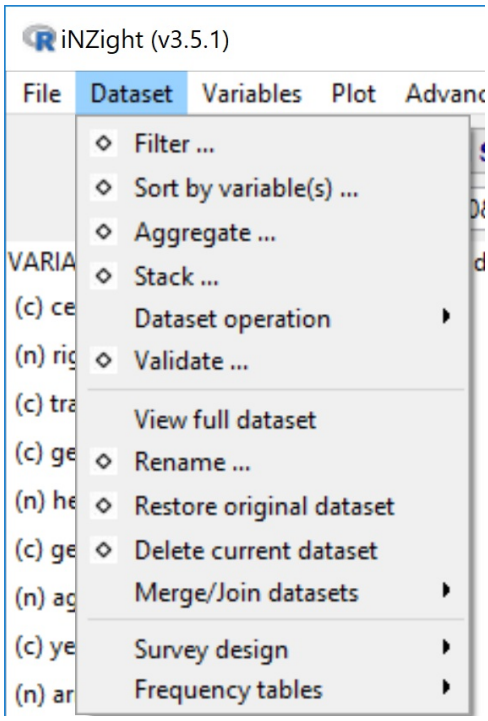
Restore Dataset

Restores the data set to the way it was when it was initially imported.

Dataset Menu

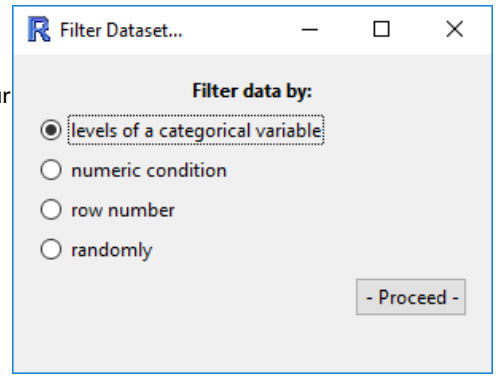
(*Organise and restructure your data, specify special structures, and filter out unwanted observations*)

Contents



- **Filter ...** (*dataset*)
  - **Levels of a categorical variable**
  - **Numeric condition**
  - **Row number**
  - **Randomly**
- **Sort (*dataset*) by variable(s) ...**
- **Aggregate ...** (*data*)
- **Stack ...** (*variables*)
- **Dataset operations:**
  - **Reshape dataset**
    - **Wide to long**
    - **Long to wide**
  - **Separate column (into) ...**
    - **into several columns**
    - **into several rows**
  - **Unite columns ...**
- **Validate ...** (*dataset*)
- **View full dataset**
- **Rename ...** (*dataset*)
- **Restore original dataset**
- **Delete current dataset**
- **Merge/Join datasets:**
  - **Join datasets** (*several **types** of join*)
  - **Append rows**
- **Survey design** (*for data from complex survey designs*)
  - **About survey data**
  - **Specify design ...**
  - **Specify replicate design ...**
  - **Post stratify ... (*or rake or calibrate*)**
  - **Remove design ...**
- **Frequency tables** (*values plus frequencies data format*)
  - Expand table
  - Specify frequency column
  - Remove frequency column

Filter Dataset

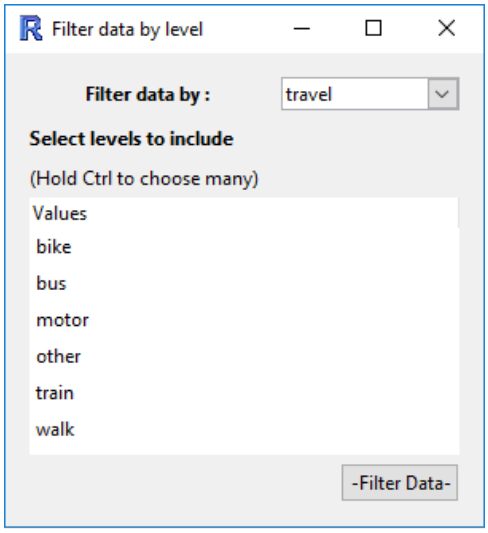


This tool provides several methods for filtering the dataset. The window that opens has four options for you to choose from:

- **Levels of a categorical variable**
- **Numeric condition**
- **Row number**
- **Randomly**

(Filter by) Levels of a categorical variable

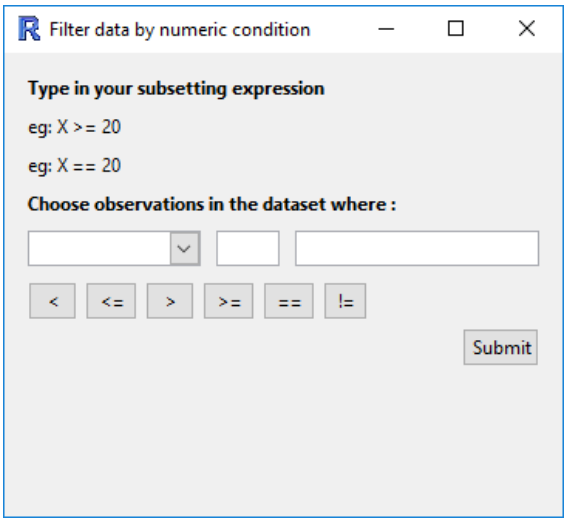
After selecting a categorical variable from the drop down box, you can select which levels you want ***to keep*** in the data set.



(Filter by) Numeric condition

This allows you to define a condition with which to filter your data. For example, you could include only the observations of `height` over 180 cm by

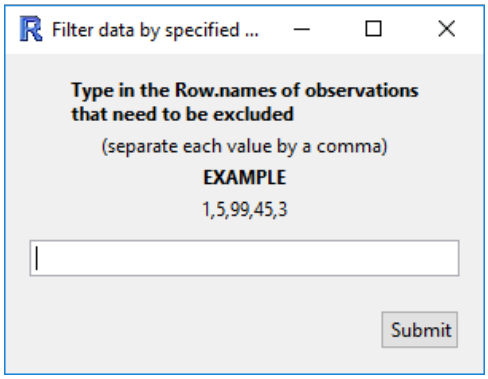
- selecting `height` from the drop down menu,
- clicking on the `>` symbol, and
- entering the value `180` in the third box.



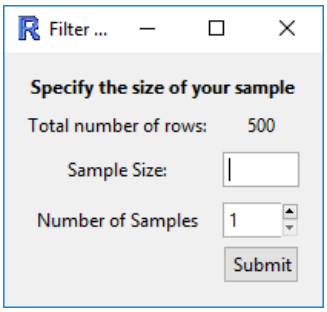
(Filter by) Row number

Exclude a range of row numbers as follows:

- Entering 101:1000 (and then **Submit**) will exclude all rows from 101 to 1000
- Similarly, 1, 5, 99, 101:1000 will exclude rows 1, 5, 99, and everything from 101 to 1000



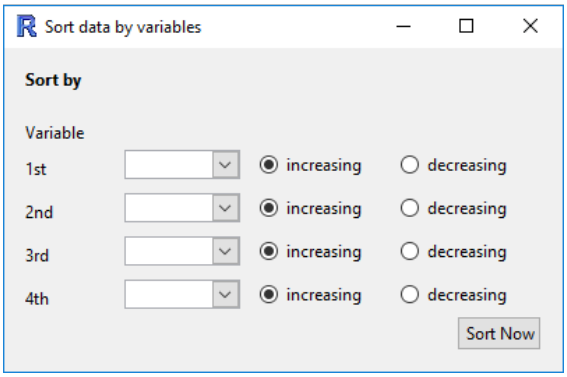
(Filter by) Randomly



Essentially, this allows you to perform bootstrap randomisation manually. The current behaviour is this:

- "Sample Size", *n*, is the number of observations to draw for each sample,
- "Number of Samples", *m*, is the number of samples to create in the new data set.
- The output will be a data set with *n* x *m* rows, which *must be smaller than the total number of rows in the data set*.
- The observations are drawn randomly *without replacement* from the data set.

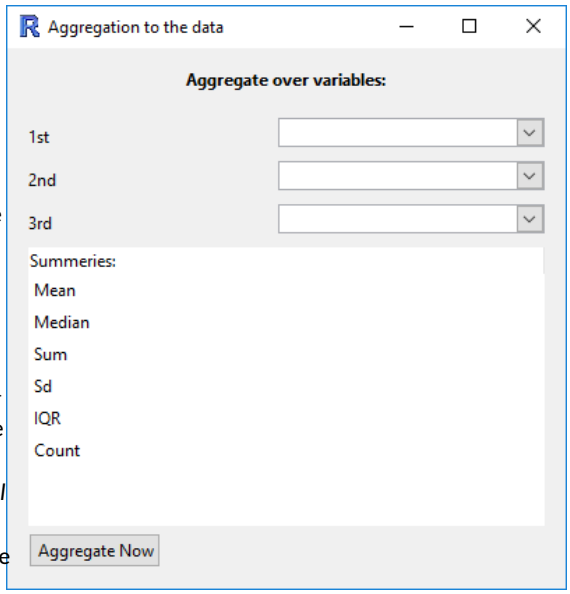
Sort data by variables



Sort the rows of the data by one or more variables

- The ordering will be nested, so that the data is first ordered by "Variable 1", and then "Variable 2", etc.
- For categorical variables, the ordering will be based on the order of the variable (by default, this will be alphabetical unless manually changed in "Manipulate Variables" > "Categorical Variables" > "Reorder Levels").

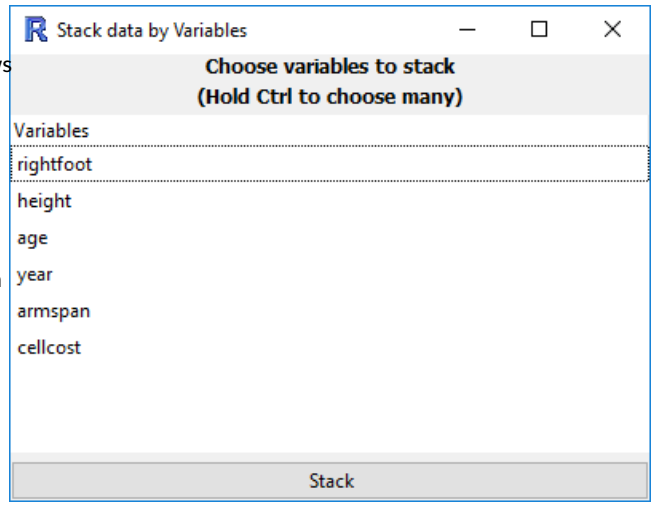
Aggregate data



This function essentially allows you to obtain "summaries" of all of the numeric variables in the data set for combinations of categorical variables.

- *Variables*: if only one variable is specified, the new data set will have one row for each level of the variable. If two (or more) are specified, then there will be one row for each combination. For example, the categorical variables `gender = {male, female}` and `ethnicity = {white, black, asian, other}` will result in a data set with 2x4 rows.
- *Summaries*: each row will have the chosen summaries given for each numeric variable in the data set. For example, if the data set has the variables `gender (cat)` and `height (num)`, and if the user selects `Mean` and `Sd`, then the new data set will have the columns `gender`, `height.Mean` and `height.Sd`. In the rows, the values will be *for that combination of categorical variables*; the row for `gender = female` will have the mean height *of the females*, and the standard deviation of height *for the females*. A visual example of this would be do drag `height` into the Variable 1 slot, and `gender` into the Variable 2 slot. Clicking on "Get Summary" would provide the same information. The advantage of using Aggregate is that the summaries are calculated *for every numeric variable in the data set*, not just one of them.

Stack variables



Convert from *table form* (rows corresponding to subjects) to *long form* (rows corresponding to observations).

In many cases, the data may be in tabular form, in which multiple observations are made but placed in different *columns*. An example of this may be a study of blood pressure on patients using several medications. The columns of this data set may be: `patient.id`, `gender`, `drug`, `Week1`, `Week2`, `Week3`. Here, each patient has their own *row* in the data set, but each row contains three observations of blood pressure.

patient.id	gender	drug	Week1	Week2	Week 3
1	male	A	130	125	120
2	male	B	140	130	110
3	female	A	120	119	116

We may want to convert to *long form*, where we have each observation in a new row, and use a categorical variable to differentiate the weeks. In this case, we would select `Week1`, `Week2`, and `Week3` as the variables in the list. The new data set will have the columns `patient.id`, `gender`, `drug`, `Stack.variable` ("Week"), and `stack.value` ("blood pressure").

patient.id	gender	drug	stack.variable	stack.value
1	male	A	Week1	130
1	male	A	Week2	125
1	male	A	Week3	120
2	male	B	Week1	140
2	male	B	Week2	130
2	male	B	Week3	110
3	female	A	Week1	120
3	female	A	Week2	119
3	female	A	Week3	116

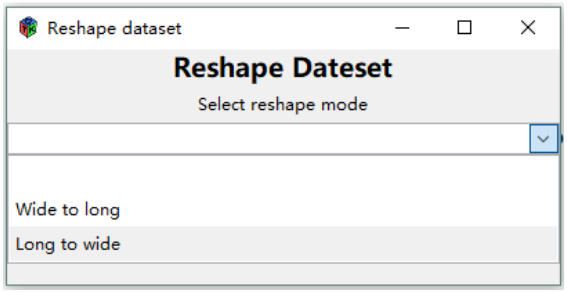
Of course, you can rename the variables as appropriate using "Manipulate Variables" > "Rename Variables".

Dataset operations

It offers three types of tools for the users to modify their dataset:

- **Reshape dataset**
- **Separate column**
- **Unite columns**

Reshape dataset

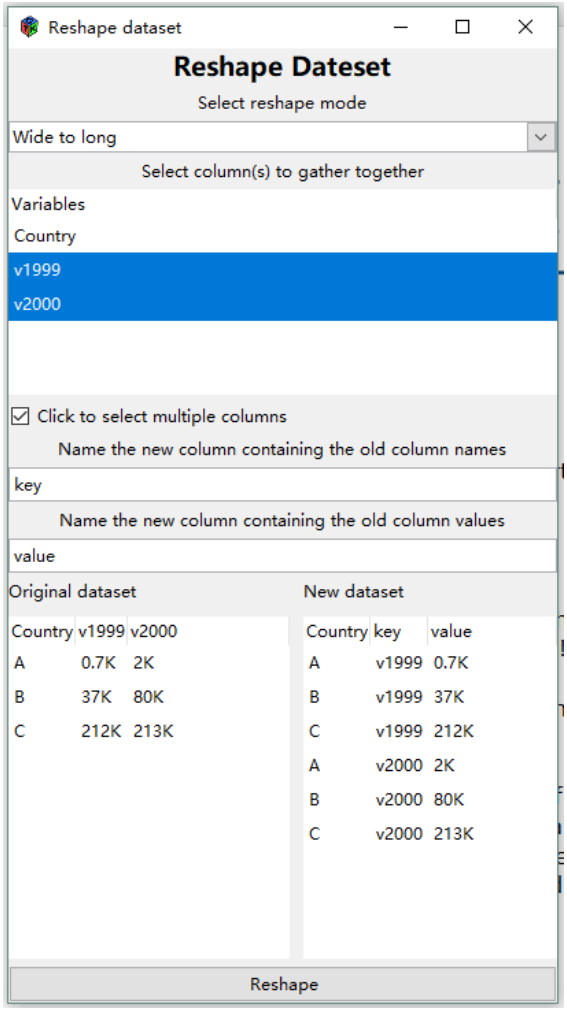


- **Wide to long**
- **Long to wide**

(Reshape) Wide to long

This allows you to select a column or multiple columns from your dataset.

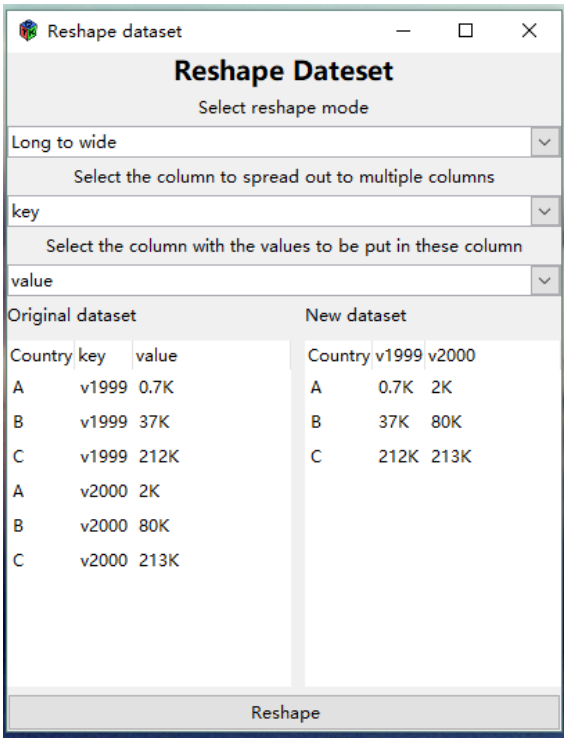
- One new column (default name `key`) is populated by the column name(s) of the selected column(s)
- The other new column (default name `value`) will contain the column value from the selected columns.
- The selected column(s) will be removed and two new columns will be added to the dataset.
- A preview panel shows what the new dataset will look like.



(Reshape) Long to wide

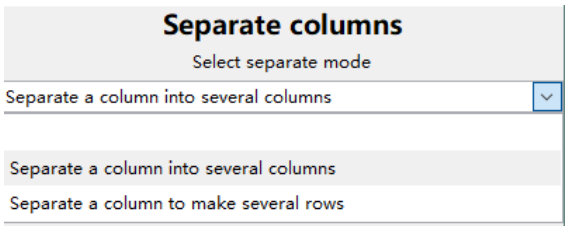
You can select a column to spread out into multiple columns (the column is named `key` in the example). It will use the column values of the selected column as a set of names for new columns.

You then select another column with corresponding values to be put into the new columns (the column is named `value` in the example).



Separate column into ...

- **Separate a column into several columns**
- **Separate a _column into several rows**



Separate a column into several columns

Allows you to separate a column into several columns using a user-defined separator.

- It will separate at every instance of the separator until no further separators are found

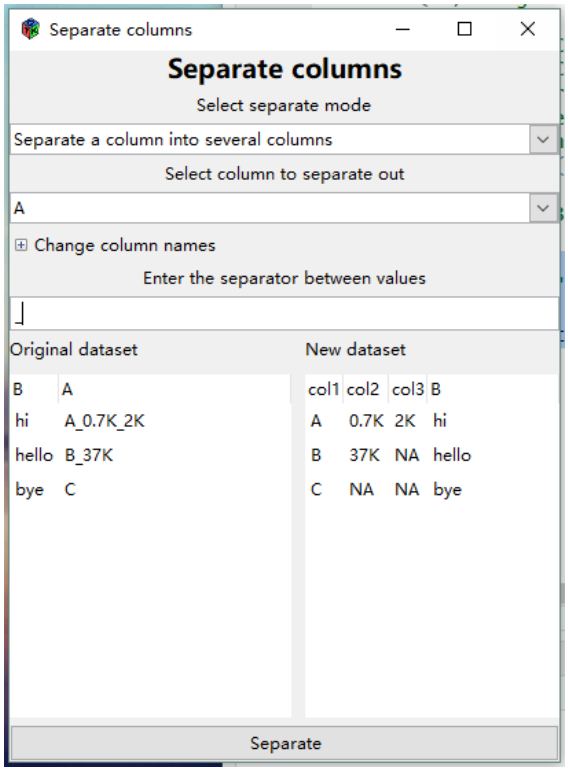
If no separator is found, the additional columns formed will contain `NA` s.

In the example on to the right, we have asked to separate column **A** using an underscore ("_") as a separator.

Because only column **A** is being separated, column **B** (or any other columns) is left unchanged in the resulting new dataset

The maximum number of fields in column **A** after separation is 3 ("A_0.7K_2K") so column **A** in the original dataset is being replaced by 3 columns with default column names (Col1, etc).

- Expanding **Change column names** (click the "+") allows you to change the default column names to something else.

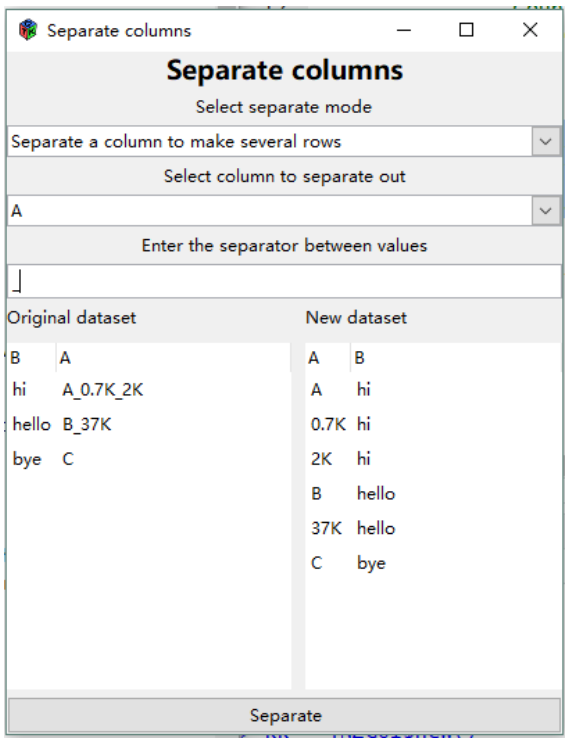


Separate a column into several rows

Instead of forming more columns this version of Separate keeps the same number of columns, with the same names, but writes more rows.

Using the same data as in the example above, the entry "A_0.7K_2K" in column **A** in the original dataset results in 3 rows in column **A** in the new dataset.

- The corresponding entries in any other columns (e.g. column **B** in the example) ***are duplicated***.

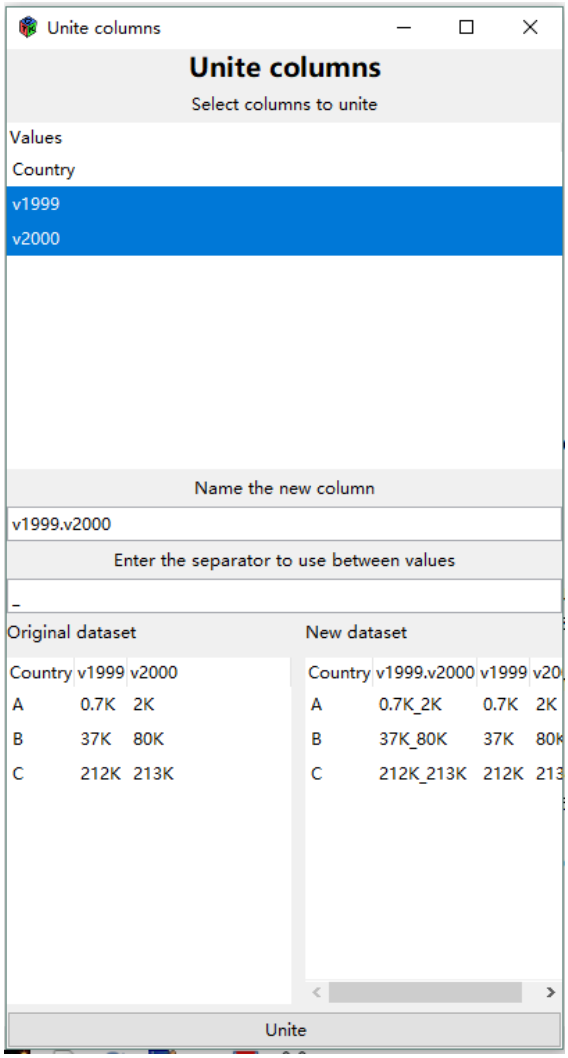


Unite columns

Allows you to select multiple columns and "unite" them using a defined separator (defaults to "_"). The united column name will be the combination of the selected columns with a "." in between.

In this example, column "v1999" and column "v2000" are united by ". The new column name is "v1999.v2000"._

- It is allowable to have ***no separator***. Just clear the ***Enter the separator*** field (delete the "_")



Merge/Join datasets

- **Join datasets**
- **Append rows**

Join Datasets

This "joins", or brings together, information in two data sets: the current dataset in iNZight and a newly imported dataset (read in using the **Import data** facility) shown at the mid-right.

Left Join: The most important joining method is called a ***Left Join***, the main purpose of which is to add new variables to the original dataset by extracting the information from the new dataset.

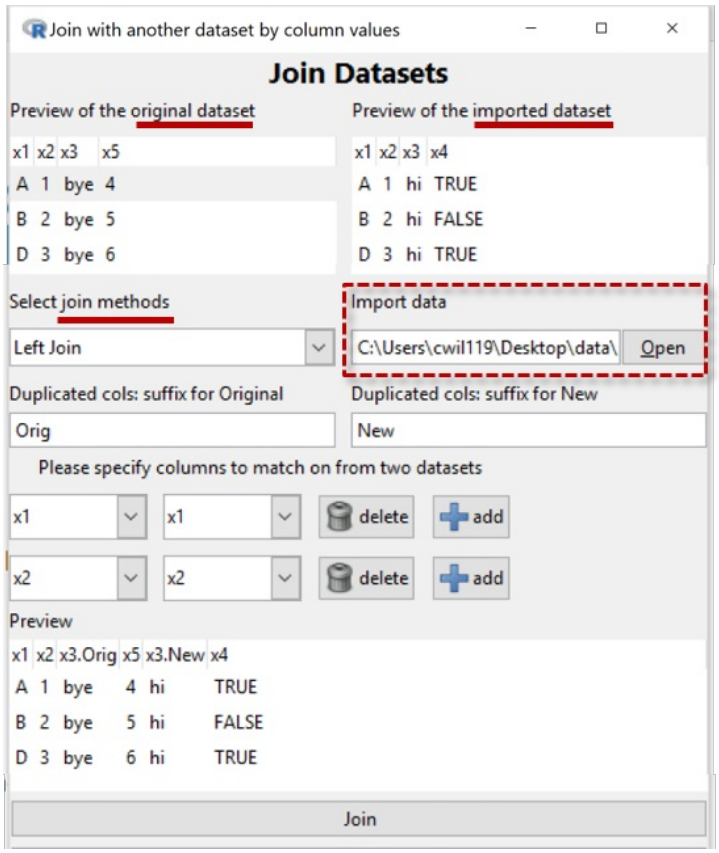
Matching rows: (***Not yet fully implemented in Lite.***) The main problem is to identify what pieces of information belong together. The most straightforward case occurs where there is a variable in the original dataset which is a unique identifier. If that variable is also in the imported dataset (even if under a different name) we can use it to match up the data which belongs to the same unit/entity.

To partially automate the process, iNZight looks for variables with the same name in both datasets (*originally x1, x2, and x3 in the Example to the right*) and offers those for determining matches.

In the Example, we have rejected x3 using the delete button beside it and so have effectively told the program that it is units with the same values of both x1 and x2 that belong together.

- The **Preview** panel shows us the effects of our choices

Click the **Join** button at the bottom once you are happy with the way the data is being joined.



The details of how the data is treated depend on the type of Join and we will **document that** after finishing describing the Example.

In the Example, x4 is a new variable so that has been added to the preview-dataset. A complication is x3 which is in both datasets but with different values for the "same" units. So the program has decided to make two variables, one for the x3 values from the original dataset and one for the x3 values from the new dataset.

Types of Join

Left Join

- The joined dataset has rows corresponding to ***all of the rows in the original dataset*** and all of its columns.
- Rows of the ***new*** dataset ***that do not have a match*** in the original dataset ***are not used***.
- The joined dataset ***also has the columns from the new dataset*** that were not used for matching.
- Rows in the original that ***have no match*** in the newly imported dataset get `NA` s for the additional columns
- **Warning:** Rows from the original dataset that have ***more than one match*** in the new dataset generate multiple rows in the joined dataset (which invalidates many simple analyses). For example, if there are 3 matches then the original (single) row will be replaced by 3 rows. The cell-values for the additional columns will be obtained from the new data set and the values for the original columns from repeating the original cell values.

How other joins differ from the Left Join

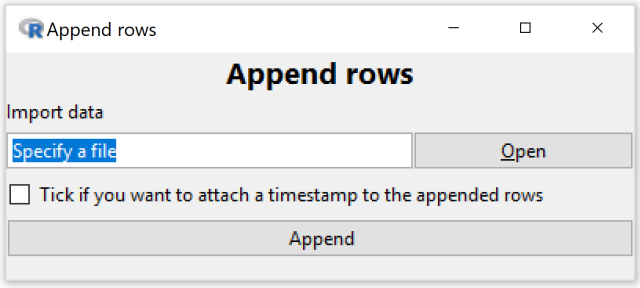
- **Inner Join:** Only use rows corresponding to matches between the two datasets
- **Full Join (Outer Join):** Also use all the non-matching rows from both data sets

[**Right Join:** iNZight does not have this. Just import the datasets in the reverse order and use a left join.]

The following are just used to filter data. No columns are added to the join from the new dataset.

- **Semi Join:** Use only rows in the original which have ***a match*** in the new
- **Anti Join:** Use only rows in the original which have ***no match*** in the new

Append rows



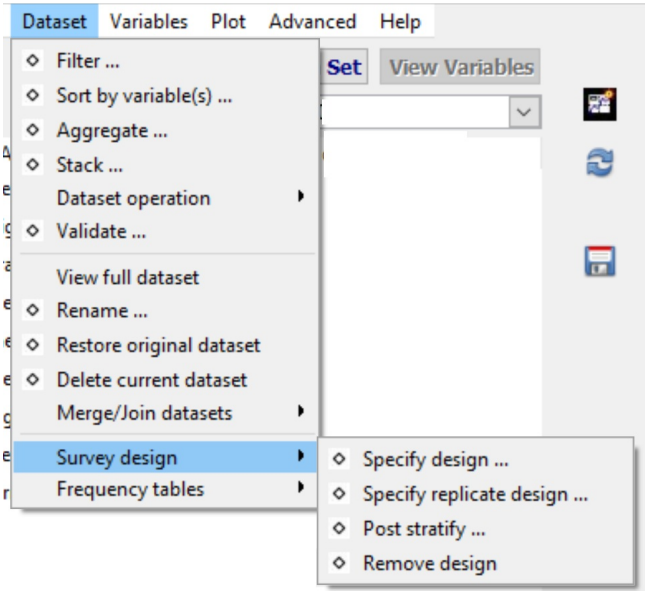
Adds to the bottom of the original dataset rows from a newly imported dataset (imported using the Import facility provided).

- The column names from both datasets are matched so the correct data goes into the correct columns
- Columns that appear in one dataset and not the other *will* appear in the result
- An error will be reported and not appending will be possible if two column names match but their types (e.g. numeric or categorical) do not match
- If the Tick-box option is selected a timestamp variable called **When_Added** will be added to the dataset recording when the new observations were added
- If there is already a variable called **When_Added** present in the original data (must be of type date-time, "(t)" in View Variables) the new timestamps will be appended to that existing variable

Where data is periodically being added to a dataset, this facility can be used to keep track of when each row was added -- thus facilitating analyses of the data as it was up until any particular time point

Survey design

- **About survey data**
- **Specify design ...**
- **Specify replicate design ...**
- **Post stratify ... (*or rake or calibrate*)**
- **Remove design ...**



About survey data

It is important that specialist methods be applied when analyzing data obtained using complex survey designs. Failure to do so typically leads biased estimates and incorrect standard errors, confidence intervals and p-values.

When a survey design has been defined almost all relevant parts of iNZight will ***apply analysis and graphics methods appropriate*** for data obtained using ***this survey design*** by applying functions in R's **survey package**.

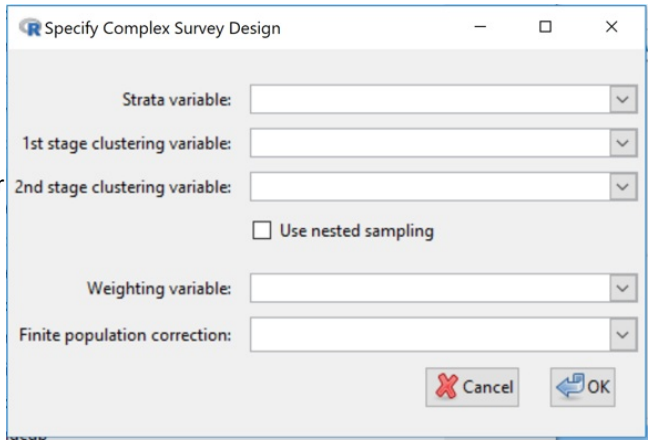
Important difference for **Get Summary**. Whereas, generally, iNZight's **Get Summary** provides summary information about the dataset itself, when a survey design is specified **Get Summary** provides estimates of population quantities -- clearly labeled as such. (Raw summaries of survey data are often meaningless because of unequal probabilities of selection.)

Regular statistical software analyses data as if the data were collected using simple random sampling. Many surveys, however, are conducted using more complicated sampling methods. Not only is it often nearly impossible to implement simple random sampling, more complex methods are more efficient both financially and statistically. These methods use some or all of *stratified sampling*, *cluster sampling* and *unequal sampling rates* whereby some parts of a population are sampled more heavily (i.e. with *higher probabilities of selection*) than others parts. These sampling features have to be allowed for in the analysis. While sometimes it may be possible to get reasonably accurate results using non-survey software, there is no practical way to know beforehand how far wrong the results from non-survey software will be.

- See **Brief introduction to survey analysis ideas**
- For an in-depth account, see **"*iNZight and Sample Surveys*"** by **Richard Arnold**.

Survey designs are typically specified to analysis programs either by **specifying a survey design** in terms of weighting, strata and clustering variables etc in the data set, or by selecting a set of variables in the dataset containing so-called **replicate weights**. **Post-stratification/raking/calibration** facilitate using additional population information (external to the survey data) to improve survey estimates. This is done after a survey design has been specified (by either method).

Specify design



Specifying a survey design in terms of weighting, strata and clustering variables etc.

iNZight's survey methods cater for simple random sampling, stratified random sampling (random sampling within each population stratum), cluster sampling and multistage cluster sampling, and complex designs involving cluster sampling within population strata.

- **Strata variable:** If **stratified sampling** has been performed, use to select the variable that specifies which stratum each observation comes from. (This variable can be either numeric or categorical.)
- **1st stage clustering variable:** If **cluster sampling** has been performed, use to select the 1st-stage clustering variable; this specifies which **1st-stage cluster** each observation comes from. (Clustering variables can also be either numeric or categorical.)
- **2nd stage clustering variable:** If two or more stages of cluster sampling have been performed, use to select the 2nd-stage clustering variable (specifies which **2nd-stage cluster** each observation comes from). Any further levels of cluster sampling (3rd stage, etc., are not used)
- **Use nested sampling:** Quite often, compilers of survey data "reuse" cluster names from stratum to stratum. Let us take, as an example, a survey in which American states for the starts and counties form the clusters. Sampled counties from Washington State may be given a County value of 1, 2, ... and counties from Arizona may also be given a County value of 1, 2, ... Clearly County 1 from Washington refers to an entirely different county from County 1 from Arizona even though they have the same value of the County variable. -- ***Click*** the **Use nested sampling** check-box if cluster labels are being recycled/reused in the data.
- **Weighting variable:** If the sampling design used unequal probabilities of selection, use this to select a variable containing the ***sampling weight*** (1 over the probability of selection) for each observation. Certain estimates will be wrong if the sampling weights do not add up to the population size, in particular estimated *population* or *subpopulation* ***totals*** and estimated *population* or *sub****population sizes***.

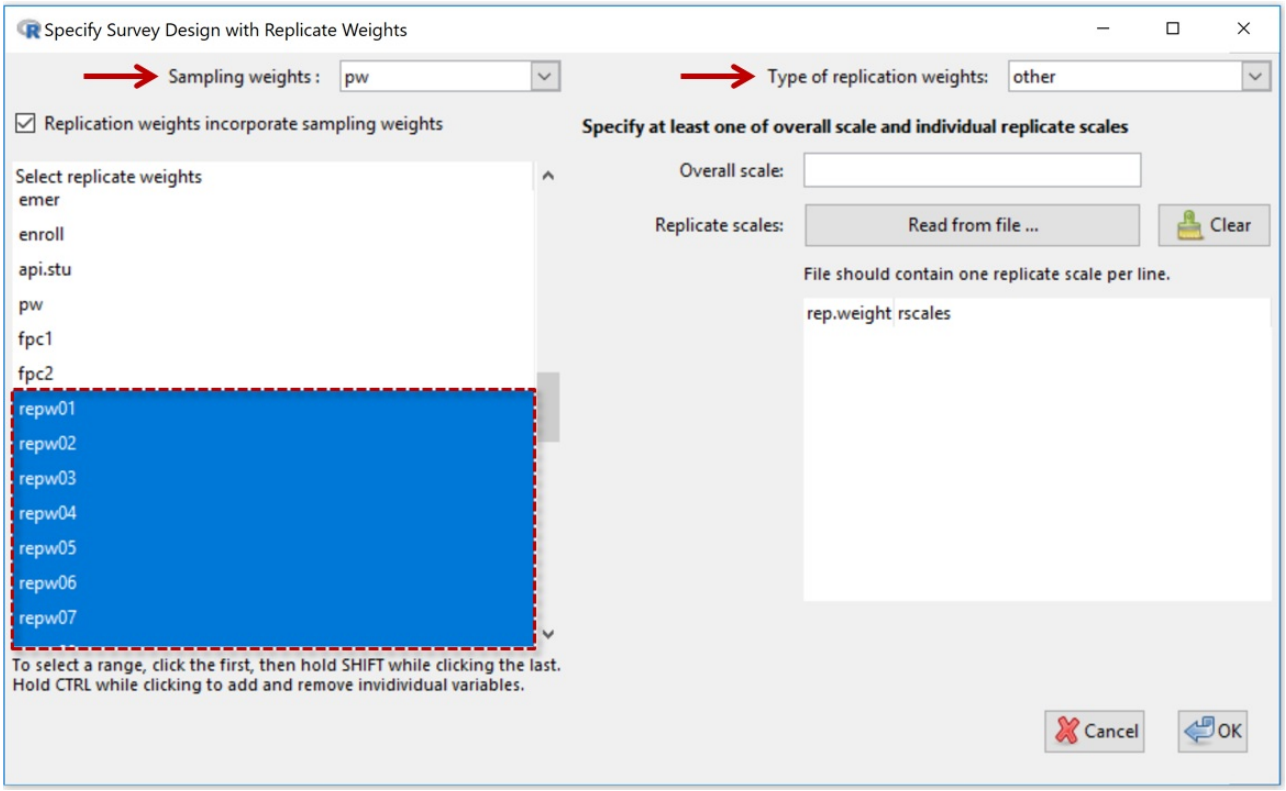
Sampling weights are often *adjusted* to allow for *unit non-response*.

- **Finite population correction (fpc):** Use this when descriptive inferences are wanted about properties of the finite population being sampled and the sample size is an appreciable proportion of the population size (e.g. > 5 or 10%). If stratified sampling has not been used, this variable should contain an estimate of the size of the population being sampled, repeated for every observation. If stratified sampling has been used, the values of this variable should contain an estimate of the size of the population stratum being sampled (differing across strata but constant within each stratum). As an alternative to using population/stratum sizes, proportions *of the total population* being sampled can be used.
  If *one stage cluster sampling has been used*(where we take a random sample of clusters, and a census within clusters), the finite-population-correction selection-box should contain an estimate of the total number of clusters (the same on every row, or the same on every row within a stratum).
  In two stage clustering we subsample within each selected cluster, and so we need to specify two variables: one for the number of clusters (just as in one stage clustering), and a second for the total number of units available for selection within each cluster (the same value on every row within the cluster). If the number of clusters is stored in a variable called **fpc1**, and the number of units within a cluster is stored in a variable called **fpc2**, then type/paste **fpc1+fpc2** in the finite population correction field.

For more information on quantities referred to in the dialog box see the documentation of the **svydesign** function in R's **survey** package.

Specify replicate design

Because making public factors like cluster membership can make it easier than survey agencies are comfortable with to identify individuals, many agencies do not distribute such information to outsiders. Instead they distribute sets of so-called *replicate weights*, slightly varying copies of the sampling weights variable that still enable survey analysis programs to make the proper adjustments to analyses of survey data.



- **Sampling Weights:** See variable selection box at the top-left
- **Replication weights incorporate sampling weights** (*checkbox*)**:** This should be ***checked if*** the replicate weights already include the sampling weights (which is usually the case). ***Uncheck this*** if the replicate weights are very different in size to the sampling weights.
- **Select replicate weights:** The large (lower) panel on the left-hand side displays the names of the variables in the dataset. Use to select the replicate-weights variables. In the example shown the replicate-weights variables were called *repw01*, *repw02*, *repw03*, ...

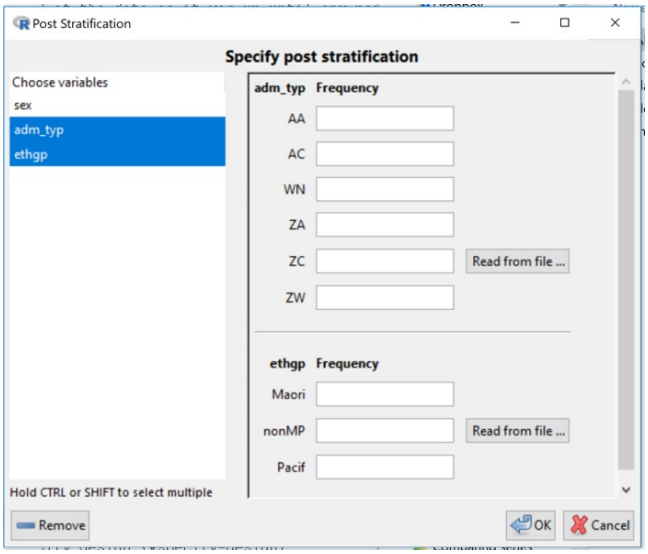
Right-hand panel

- **Type of replication weights:** Depends on the type of replicate weights the person who compiled the dataset has used. Select from list - *BRR*, *Fay*, *JK1*, *JKn*, *bootstrap*, *other*.
- **Overall scales:** Only used for Types *bootstrap* and *other*.

For more information on quantities referred to above

- see the documentation for the **svrepdesign** function in R's survey package and also **here**
- see also the **section on replicate weights** in "***iNZight and Sample Surveys***" by **Richard Arnold**..

Post stratify



This allows for ***poststratifying/raking/calibrating*** a design that has already been defined using either of the methods above above. (Technically the design is updated using the **calibrate** function in R's survey package.)

This allows a data-analyst to improve estimation by augmenting the information in the survey data by adding information on the whole population where this is available from other sources. Categorical variables in the data set are offered as possible poststratification/raking/calibration candidates. Corresponding population counts can be input by typing or reading from files.

In the example in the screenshot, there are 3 categorical variables in the data set offered as possible candidates and 2 have been selected. The design would then be calibrated using user-supplied information on population counts for the all the categories of both **admin_type** and **ethgp**.

This is fine when you only want to use information about single variables, but what if you have population information on the cross-classification (all possible combinations) of **admin_type** and **ethgp**, say? Then you would have to create a new variable in the dataset, called say **admin_type.ethgp** that has all these combinations. This can be done with **Variables > Categorical Variables > Combine categorical variables**.

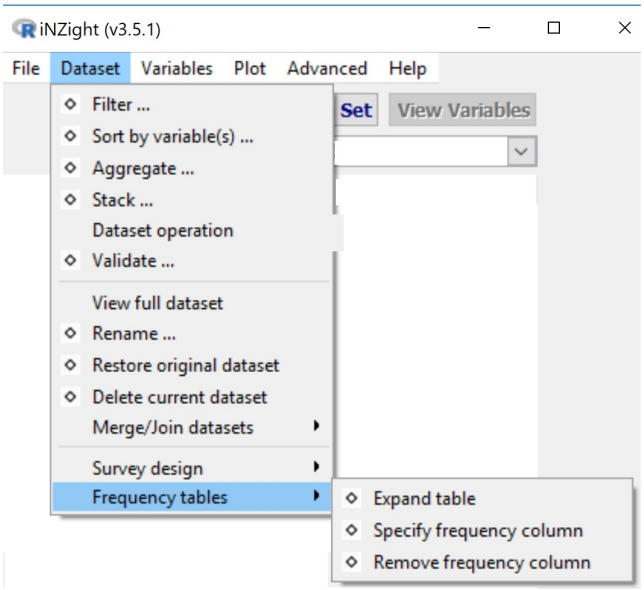
***Warning*:** Currently the new variables have to be set up before specifying a survey design. If you only think about it later you will need to use **Remove design** (next item), set up new variables and then re-specify the design.

For more information on quantities referred to in the dialog box **see here ....**

Remove design

Discard design information and ***revert to using standard methods of analysis***.

Frequency tables (*values plus frequencies data format*)



When a **frequency column** is specified, graphics and summaries respond to a combinations-of-variables-plus-their-frequencies data structure. Only categorical variables are retained.

The **expand table** option can be used to work with both numeric and categorical variables. The storage and efficiency savings of working with a unique combinations of variables plus frequencies will typically be very much smaller here.

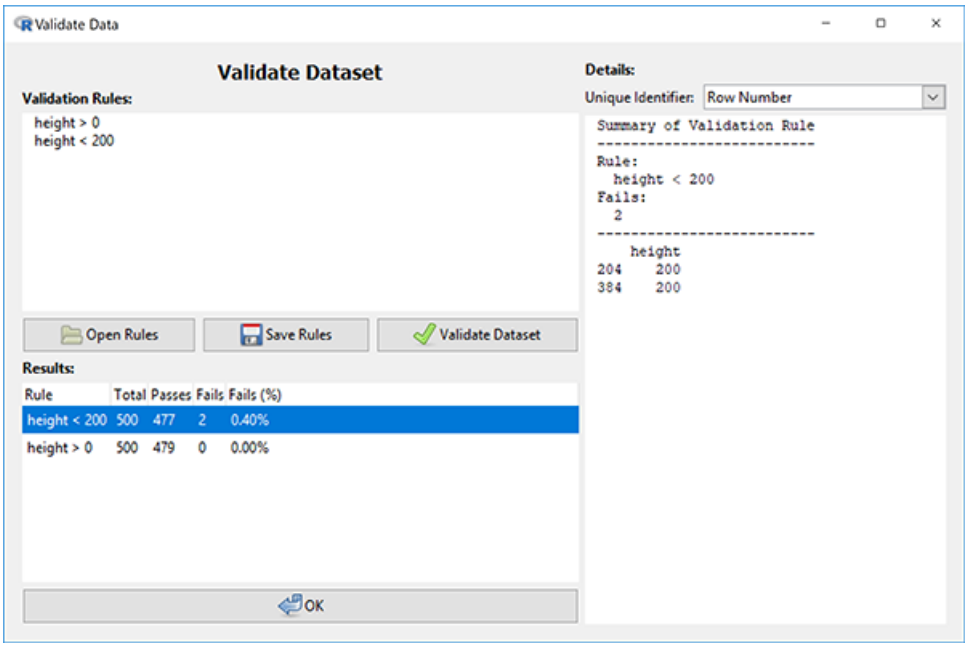
Validate Dataset

Often, we want to validate that the data in a dataset adheres to our expectations about how that data should behave based on external real world-knowledge.

- ***Validating a data set necessarily requires user-supplied validation rules.***

For example, we expect the heights of humans are not negative or someone cannot work for more than 24 hours in a single day.

The validation window gives us the ability to define or import rules, and check whether the data conforms to those rules identifying any exceptions.



- In the validation window, rules can be ***typed*** into the **Validation Rules** text box in the top left ***or imported*** from a text file using the **Open Rules** button.

Using the first example, the rule to check whether heights are above 0 can be written in this textbox as `height > 0` in this text box (given there is a variable named `height` in our dataset).

- **To check all of the rules**that have been defined in this textbox, ***click*** the **Validate Dataset** button.

The results of each rule are presented in a table at the bottom of the window and show the number of observations that were checked ("Total"), the number of passes and failures ("Passes" and "Fails" respectively), and the fail percentage ("Fails (%)"). Initially this table is sorted by failure percentage, but you can click on other column headers to order the list in other ways.

- The **Unique Identifier** selection box at the top right of the validation window allows you to select the name of a variable that contains unique identifiers for the units/cases/rows in the data set.

This is more useful than employing row numbers (the default setting) because unique-identifier values remain unchanged in the data when rows are deleted whereas row numbers will change.

- **Double clicking on a row*of the results table*** will generate a ***detailed breakdown*** of the results in the **Details** section on the right-hand side of the window.

This breakdown will provide details about the observations which failed on that particular rule, giving the row numbers (or unique-identifier values if a unique-identifier has been selected) of these observations and the values used to assess the rule.

Validation rules or changes to imported rule files are ***discarded*** once the validation window is closed.

- If you would like to store the set of rules you have defined or save any changes to an imported rule file, this can be done using the **Save Rules** button.

They are saved into a text file on your computer that can be imported again using the "Open Rules" button or viewed using a text editor.

The rules you use to validate the dataset do not need to be simple comparisons between a variable and a static value as in the previous example. ***More complex rules*** can be built by performing calculations on the variables, e.g. `weight / height^2 < 50` will verify that each observation's body mass index is below 50. The values of each variable contained in the calculation as well as the end result are provided in the detailed breakdown.

- Instead of comparisons between a variable/calculation and a static value, we can ***compare against*** another variable or ***a calculation based on the data***.

For example, to check that the income of an individual (contained in variable `Income`) is no more than 1000 times their number of hours per week (contained in variable `Hours`), we can use the following rule: `Income <= Hours * 1000`. This will calculate a different value to compare income against for each observation.

For more information on what is possible using validation rules, the vignettes and help files of the underlying R package (validate) might be useful: **Introduction to Validate vignette** and **Validate package (in particular, the syntax section of the reference manual)**

Restore Dataset

Restores the data set to the way it was when it was initially imported.

Variables Menu

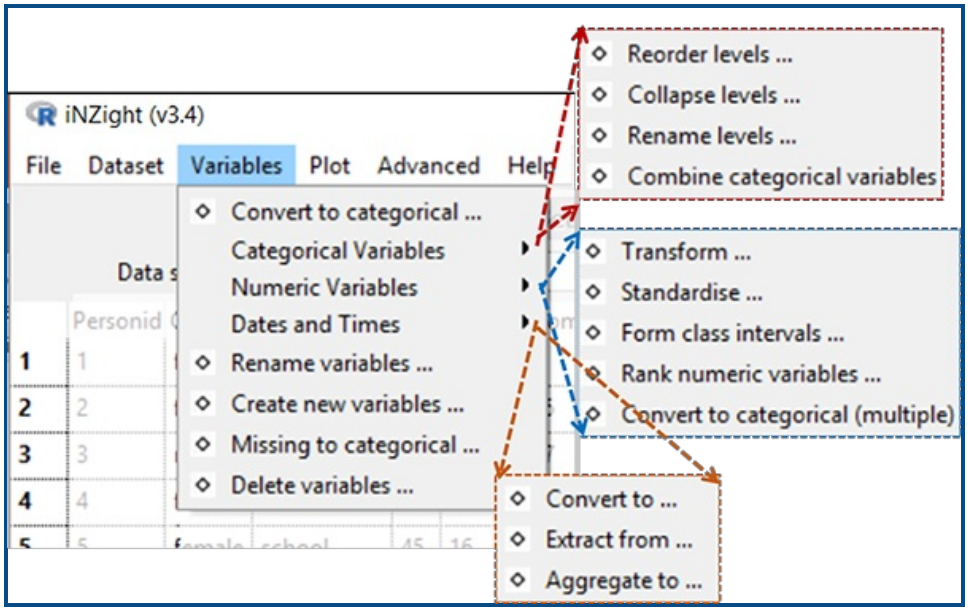
(*Called **'Manipulate Variables'** in iNZight Lite*
Convert variables to new types, create new variables, and rename existing ones)

iNZight assumes that data sets are in rows = cases by columns = variables format. For example, the cases (also often called units) may be individual people and the columns = variables contain different types of "measures" on those people.

By default, if **all the values** of a variable are **numbers**, then that variable will be treated as a **numeric variable**.

If **any of the values** contains even one **alphabetic character**, then (by default) the whole variable will be treated as a **categorical variable** (i.e., one that gives group membership). The one exception is the value `NA`, which is treated as a missing-value code in both numeric and categorical variables.

Page Contents:



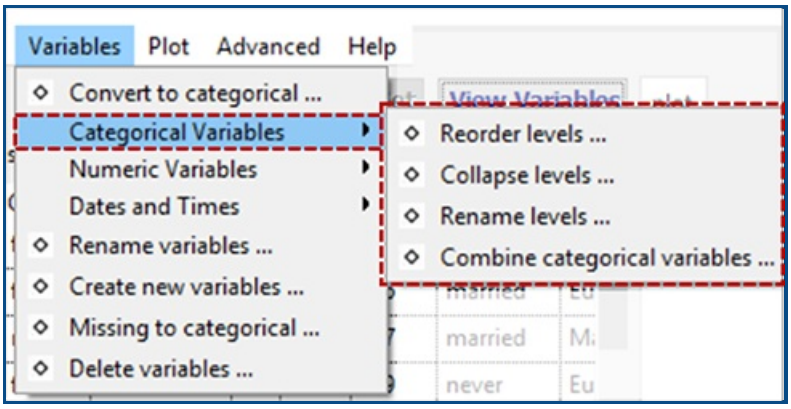
- **Convert to Categorical ...**
- **Categorical Variables**
  - **Reorder Levels ...**
  - **Collapse Levels ...**
  - **Rename Levels ...**
  - **Combine Categorical Variables ...**
- **Numeric Variables**
  - **Transform Variables ...**
  - **Standardise Variables ...**
  - **Form Class Intervals ...**
  - **Rank Numerical Variables ...**
  - **Convert to Categorical (Multiple) ...**
- **Dates and Times**
  - **Convert to ...**
  - **Extract from ...**
  - **Aggregate to ...**
- **Rename variables ...**
- **Create new variables ...**
- **Missing to categorical ...**
- **Delete variables ...**

Convert to Categorical

Creates a categorical version of a numeric variable (for more sophisticated version see ... **Convert to Categorical (Multiple) ...**).

Categorical Variables

Reorder Levels



By default, the levels of a categorical variable are displayed in alpha-numeric order. This enables you to change from the default order of display to something more natural; e.g. from {"adolescent", "adult", "child", "elder"} to {"child", "adolescent", "adult", "elder"}.

Collapse Levels

Combine levels within a categorical parent variable to make a new variable with a smaller number of levels. (The levels of a categorical variable are the set of unique, or distinct, values it takes).

Rename Levels

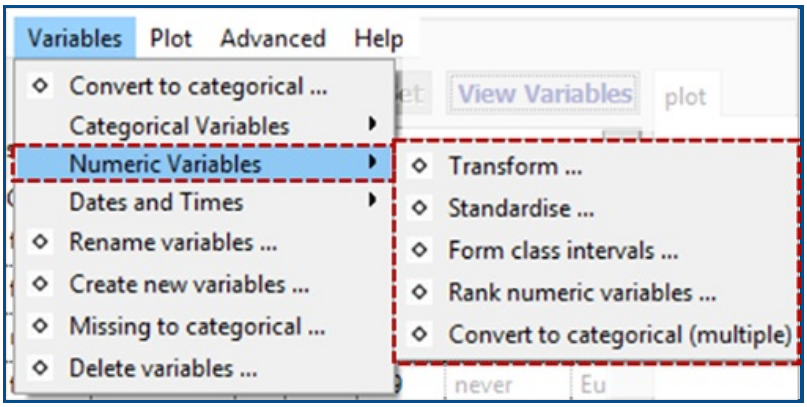
Change the names of the levels of a categorical variable, e.g. from (ages) {"< 12", "12-17", "18-69", "> 69"} to {"child", "adolescent", "adult", "elder"}.

Combine Categorical Variables

Take two categorical variables and create a new categorical variable whose levels are all combinations of those two (e.g., the combinations of ethnicity and gender).

Numeric Variables

Transform Variables



Creates a new variable that is a transformed version of the parent variable. Transformations available are `log` (base `e` or base 10), exponential, square, square root and reciprocal.

Standardise Variables

Create a **standardised** version of a variable (or **z-score**) by subtracting the mean of the variable from each value and dividing by its standard deviation.

Form Class Intervals

Create a categorical variable whose levels are class intervals of a numeric variable. For example, take `age` and create `age.f` with levels {" {0,20]", "(20,60]", "(60,110]"}.

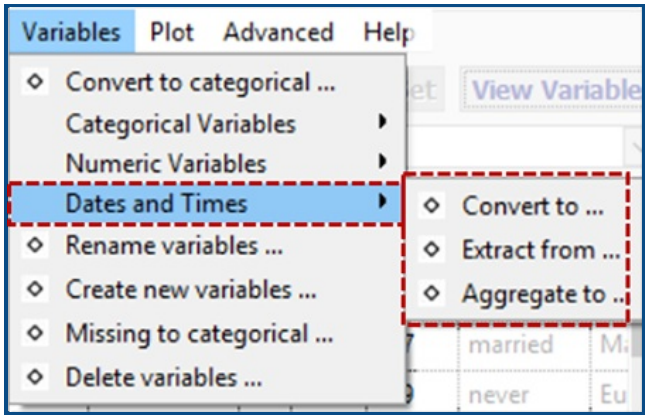
Rank Numerical Variables

Creates a new numeric variable containing the *rank*, or *order* of the selected variable. This is often used when the order of the values (but not the values themselves) is of interest.

For example, the variable `age = {15, 12, 19, 16}` would result in the new variable `age.rank = {2, 1, 4, 3}` .

Convert to Categorical (Multiple)

This has the same result as the previous "Convert to Categorical" tool, however it allows users to convert multiple variables at once.



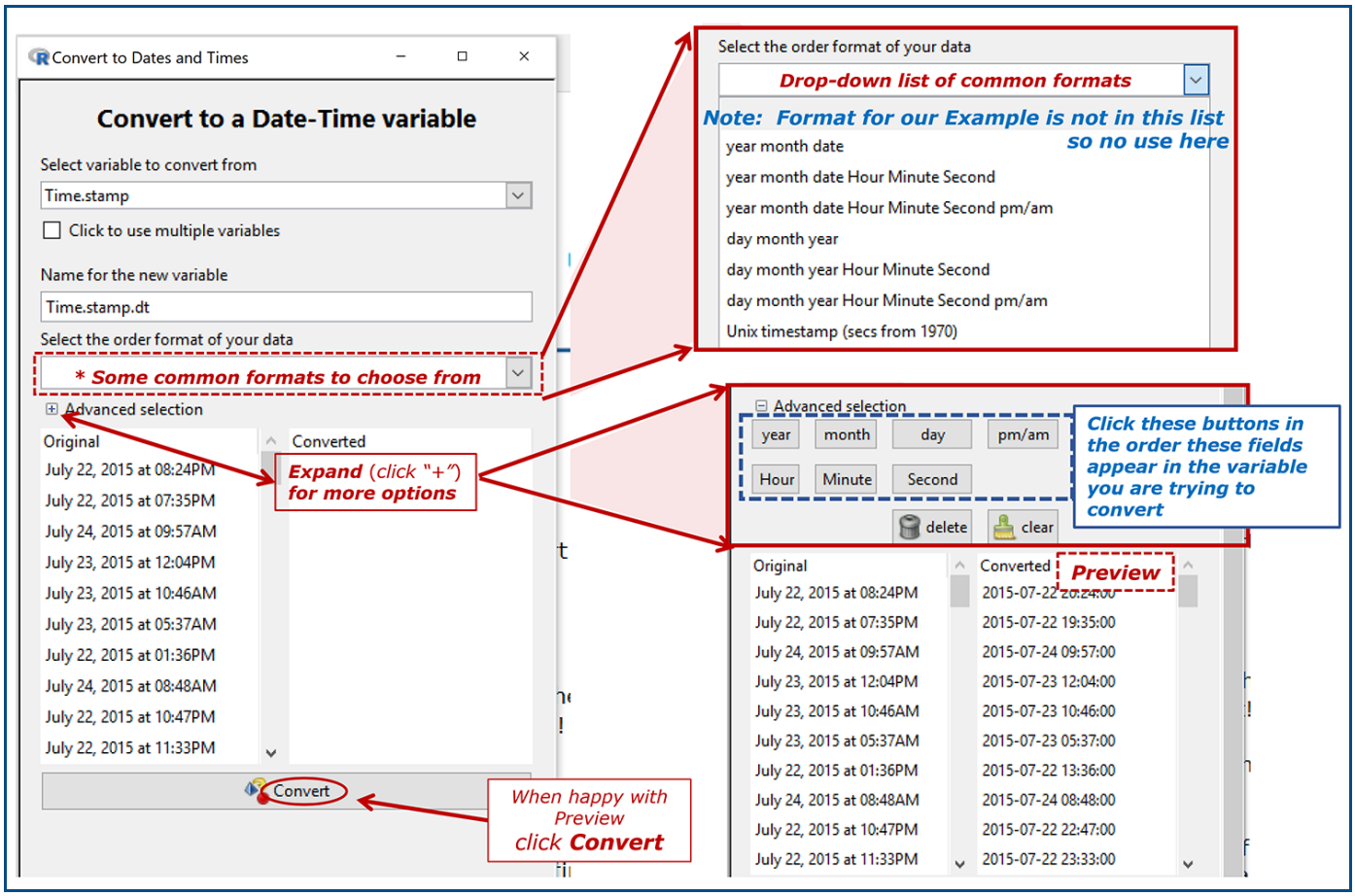
Dates and Times

- **Convert to ...**
- **Extract from ...**
- **Aggregate to ...**

(Date-times) Convert to ...

Convert a variable to a standard dates and times (POSIXct) format and ***saves*** as a ***new variable***.

In the example that follows, Time.stamp *is a categorical variable that we wish to convert to a variable that iNZight recognises as a datetime variable.*

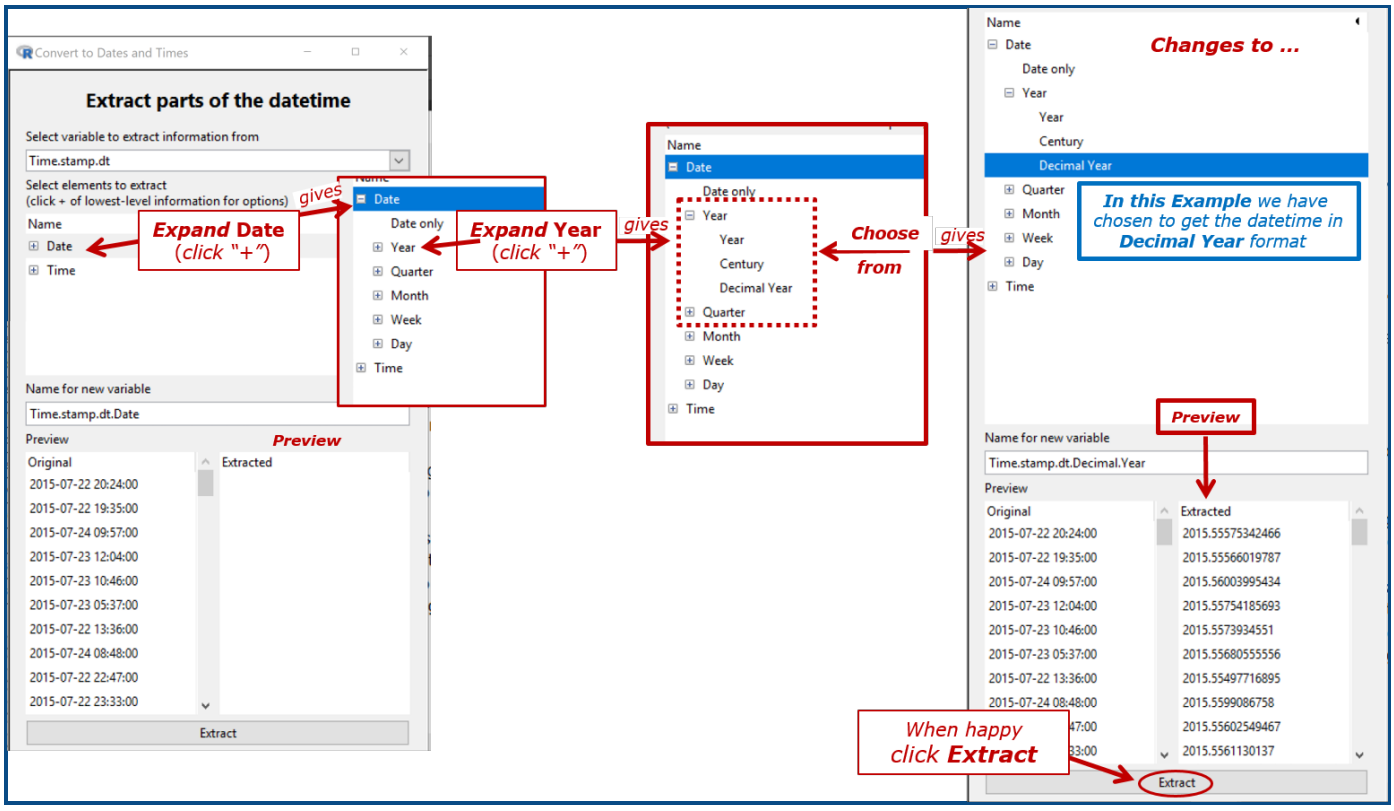


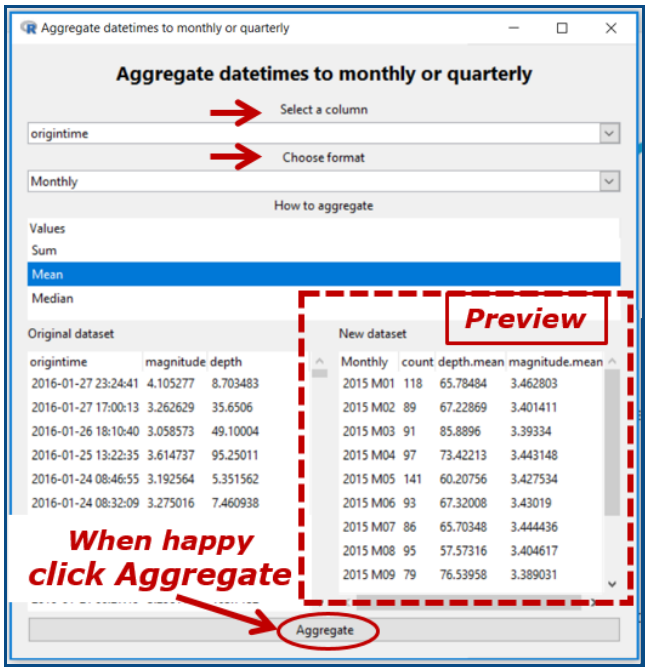
(Date-times) Extract from ...

Creates a new variable by extracting specific component (e.g. data, year, time...) from a dates and times variable in a large variety of levels of detail and of format.

- The pattern is to ***expand the option*** (click the "+") referring to the ***most detailed level of information*** you want in the result. The preview will show you what the result of your choice will look like.

In the example that follows, Time.stamp.dt *is a categorical variable that we wish to convert to a variable that iNZight recognises as a datetime variable* Timestamp.dt. *What we want to do is extract a new variable containing the Year and the decimal part of the Year ("Decimal Year")*





(Date-times) Aggregate over ...

Converts to

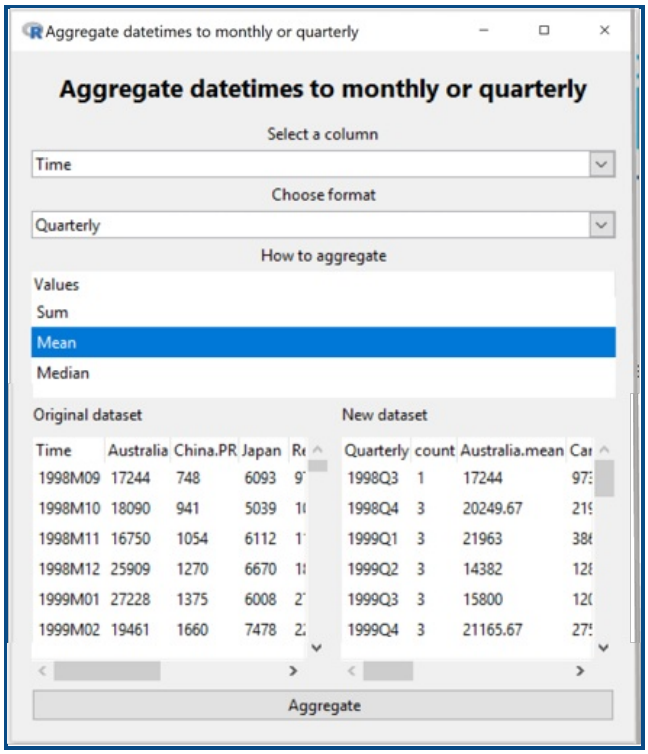
- a ***weekly*** (*e.g. 2019 W7*), ***monthly*** (*e.g. 2019 M5*), ***quarterly*** (*e.g. 2019 Q3*) or ***yearly*** (*e.g. 2019*) variable in the form that the time series variable likes

and aggregates using ***sum***, ***mean***, or ***median***.

In the Example, `origintime` *is a date-time variable* [thus, annotated `(t)` in View Variables] *and we have chosen to convert it to monthly* (`yyyy Myy`) *with all of the values in the same month aggregated to their* `mean` .

The `count` column shows how many values have been aggregated. If there are missing values present in a variable being aggregated, then another column of the form `variable.missing` . Missing values are ignored when aggregating.

In the Example below a categorical variable in monthly (`yyyy Mzz`) *format is aggregated to quarterly* (`yyyy Qz`).

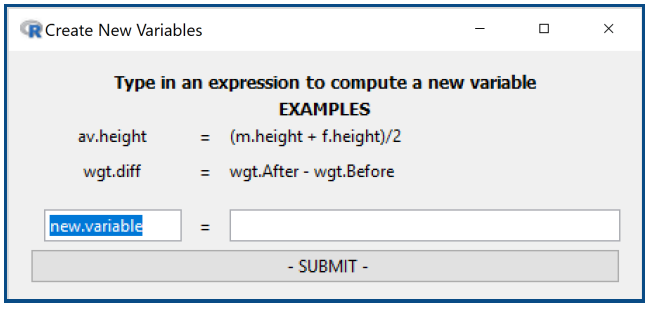


Rename Variables

Rename variables in the dataset. Especially useful for variables created by iNZight.

Create New Variable

A very flexible facility for creating new variables from existing variables, essentially by doing arithmetic on them.



Left-hand box: desired name for new variable
Right-hand box: any valid **R** expression

Examples

- `income = hours * payrate`
- `weight.diff = end.weight - begin.weight`
- `average.weight = ( begin.weight + endweight) / 2`

Missing to Categorical

- **Categorical variables**: Any missing observations for the variable we be given a new level, `missing`. All others remain the same.
- **Numeric variables**: The variable will be converted into a categorical variable with two levels: `missing` and `observed`.

Delete Variables

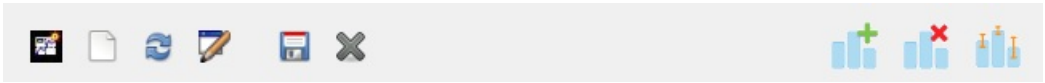
Delete variables from the dataset. Click variable names using the **Shift** and **Control** keys to choose multiple variables for deletion.

iNZight User Guides: Plot Options

iNZight provides users with several different tools for adjusting the appearance of plots. Change colours, plot types, or point size; add trend lines, or confidence intervals.

Table of Contents

Plot Toolbar Buttons



The plot toolbar lies below the graphics window, and provides several useful options to users. The buttons are described below, from left to right.

New Plot Window

Clicking this button will open up a new graphics window that uses the default R device instead of the "container" used by iNZight. This has two particular uses:

- To permit copying and pasting of graphics (including in metafile form) by right-clicking on the graphics window,
- To permit easier resizing of the plotting window, including full screen.

New Plot Tab

This will create a new plot tab, allowing the currently displayed plot to be retained. Subsequent plots will use the new tab. You can switch between tabs by *double-clicking* on the tab you wish to plot into.

Redraw

Use if the plot window seems to have stopped working or is not displaying your plot properly. It re-draws the data and usually corrects the error.

Rename Plot Tab

Used to rename the tab of the current plotting window.

Save Plot

Enables you to save the contents of the displayed plot as a file (in `.jpeg`, `.png`, `.bmp`, `.tiff`, or `.pdf` formats).

Close Tab

This will close the current **plot tab**, *not the iNZight window.*

Add to Plot

Remove Additions

For removal of some or all additions made to a plot.

Inference Information

Add to Plot

The Add to Plot window lets your customise your graph, explore patterns in the data, and highlight important features. The options available to you depend on the type of variables you have selected and the plot drawn.

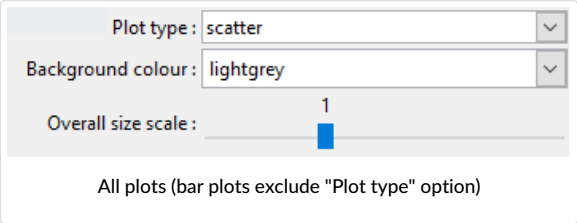
At the top of the Add to Plot window is a drop down that lets you select from one of the available panels:

- **Customise Plot Appearance**
- **Trend Lines and Curves** (scatter/hexbin/grid-density plots only)
- **Axes and Labels**
- **Identify Points**

Customise Plot Appearance

This panel allows you to control most of the visual aspects of the graph, including size and colour.

General Appearance



This section is common across all graphs, with the exception of the options listed in the "Plot type" dropdown.

- **Plot type**: allows overriding the default plot chosen by iNZight, which is based on dataset size.

  Available Plot Types

- **Background colour**: you can customise the background colour of your graphs to suit your preference, or make certain features easier to distinguish.

  You can **specify colours** using names or HEX codes.

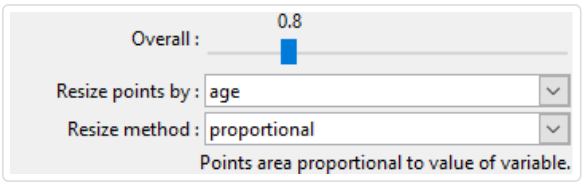
- **Overall size scale**: this lets you adjust the overall size of everything drawn on the screen.

  This works as a baseline, so all other size settings will be multiplied by this value.

Size / Point Size

Control point sizes on scatter plots and dot plots, hexagons on hexbin plots, and bar-widths on histograms.

Size by variable: On scatter plots, you can choose a numeric variable to be used to resize the points.



Colour / Point Colour

Change the colour of points, hexagons, and bars. Colours can be chosen from the list, or you can type your own. **See choosing colours** for more information.

Colour by variable: (scatter plots / dot plots / hexbin plots / one-way barplots).
Select a variable to code colours. Several choices for colour palettes will be offered.

Read about iNZight's palettes

- **Categorical variable**: Each category will be assigned a colour (depending on your chosen palette)

  Scatter plots and dot plots: Points are coloured according to the level of the variable.

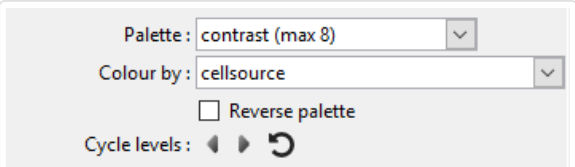
  Hexbin plots: hexagons are coloured depending on the proportion of points in each category that fall within that hexagon.

  One-way bar plots: the bars are segmented depending on the proportions of each category and coloured accordingly.

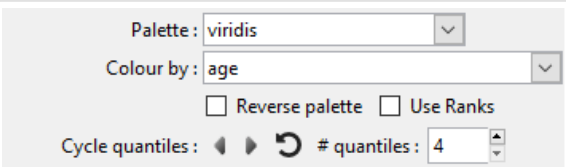
- **Numeric variable**: a linear scale is set up from the lowest to the highest value, and each point is coloured based on its value.

  If you click the **Use rank** checkbox, the colours will be based on quantiles, rather than absolute values. This can be helpful if you have a few very large values of the chosen variable.

  *NOTE: if you choose a numeric variable on a **hexbin plot**, iNZight will convert it to a 4-level categorical variable.*



Colour by a categorical variable
Dot / Scatter / Hex / One-way Bar



Colour by a numeric variable
Dot / Scatter

Cycle levels: You can cycle through the various levels (or quantiles for numeric variables), making it stand out from the others, by clicking the left and right arrows. You can adjust the number of quantiles to use by adjusting the number in the box.

Point Symbols

Select different symbols to use on scatter plots and dot plots.

Symbol by variable: You can also chose a categorical variable with 5 or fewer levels, and iNZight will give each its own symbol. This is particularly useful when creating colour-blind friendly plots, or for black-and-white printed graphs.

WARNING: transparency can make drawing very slow if you have a large dataset

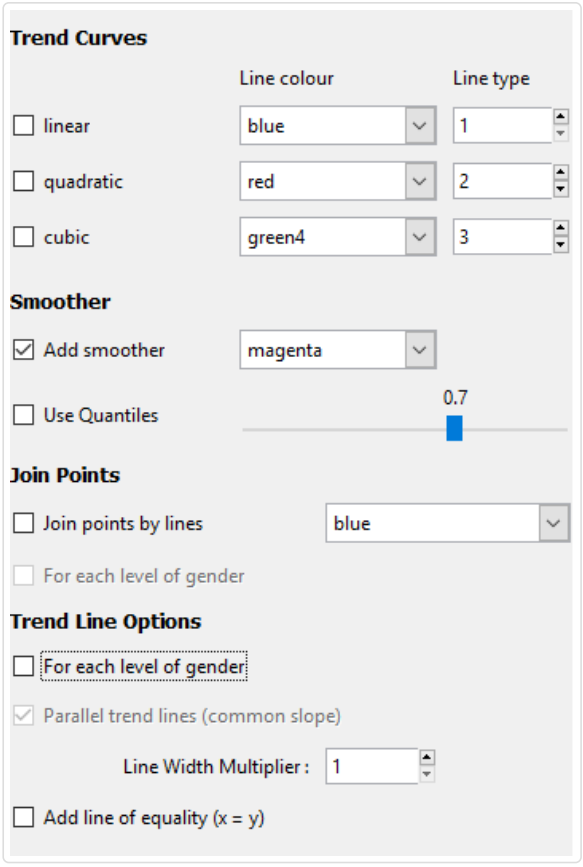
Diamonds and triangles can take a long time to draw if you have a big data set (at least on Windows). Circles (the default) and squares seem to be OK.

iNZight will turn off transparency if you change plotting symbol to one of these; you can put transparency back, but iNZight may become unresponsive while it tries to draw the graph.

Trend Lines and Curves

On *scatter plots*, *hexbin plots*, and *grid density plots*, you can add various types of lines to the graph.

Trend Curves



Trend curves are fitted using **linear regression** models, which minimises the overall vertical distance between points and the line.

The formula for the line (which can be found by clicking the **Get Summary** button after adding a trend line) depends on the type of curve fitted. In these equations, y is "Variable 1", the *primary variable of interest*, and x is "Variable 2". The greek letter β ("beta") represents an **unknown value**, and iNZight picks the best value to make the line fit the data as well as possible. The *subscripts* ($\beta_0, \beta_1, \ldots$) simply indicate **different** unknown values.

- **Linear**: a straight line fitted through the points:

  $$y = \beta_0 + \beta_1 x$$

  It has an **intercept**, β_0, which represents the value of y when $x = 0$, and a **slope**, β_1, which describes the change in y for a unit change in x.

- **Quadratic**: a curved line with at most one bend:

  $$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

- **Cubic**: a curved line with up to two bends:

  $$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

For more details, **see this page of the Data to Insight course**

Smoothers

These curves are fitted without the restrictions of the regression models shown above. iNZight uses a **loess** smoother, and you can control the degree of "smoothness" using the slider.

If you have a large data set, you can add **quantile smoothers**, which draw lines at not only the middle (median) of the data, but also at the quartiles, 25% and 75%, and the 10% and 90% quantiles if the sample size is large enough.

Join Points

In some cases, you may have **ordered data** that makes sense to connect points, for example a time series. If you specified a categorical **colour by** variable, you can optionally connect points within each level of that variable.

Trend Line Options

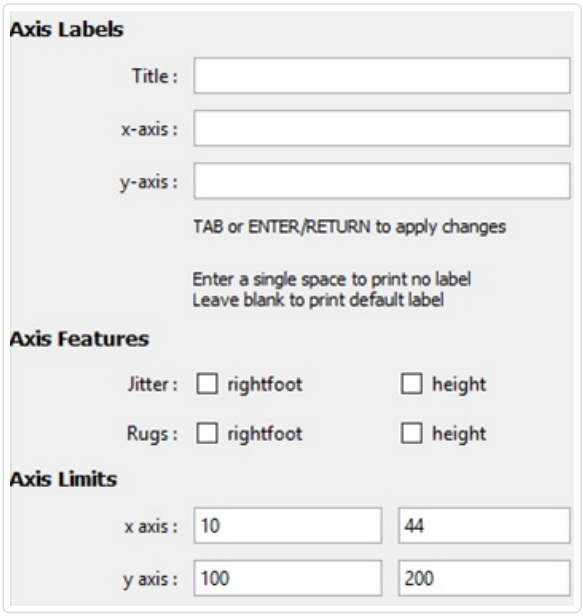
If you specified a categorical **colour by** variable, you can fit a trend curve through each level of that variable. By default, the lines will have the same **slope**, but different **intercepts**, so they will be parallel. If you want the lines to be completely independent, you can uncheck the parallel trend lines box and each level will also be given its own intercept.

Line Width Multiplier will adjust the thickness of lines.

The line of equality is useful when the units of the two variables are the same (for example, "before" and "after" measurements).

Axes and Labels

Axis Labels



Type your own labels to override those automatically created by iNZight.

To remove labels, just enter one or more spaces into the box and press `enter`.

Axis Features

For **scatter plots** only.

- **Jitter**: This adds a small amount of noise to each value, which helps to separate out discrete variables (for example, age).
- **Rugs**: This adds a small line on the axis for each point, making it easier to read values for extreme points.

Axis Limits

This allows you to adjust the limits of the axes, effectively allowing you to "zoom in" on specific regions.

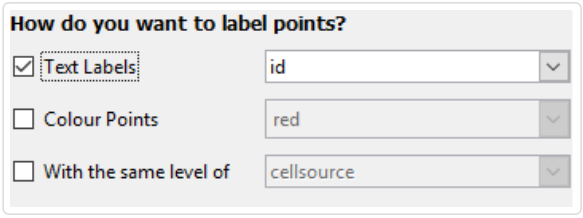
Number of bars

On bar charts, you can adjust the total number of bars shown at a time. This is useful if a categorical factor has too many levels to display all at ones.

Identify Points

This allows users to label points in one of three ways (text labels, colour labels, and related points), using one of three methods (clicking, selecting values, or labelling extreme points).

How do you want to label points?

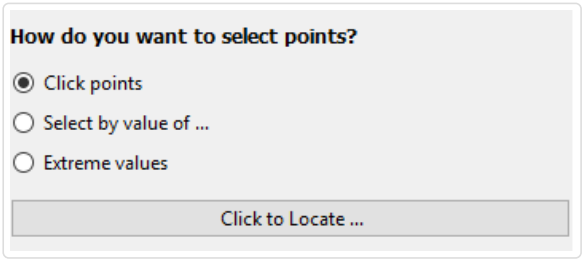


- **Text labels** allows you to select a variable from the data set with which to label selected points
- **Colour points** allows selected points to be filled in with a selected color. (See **ways of choosing colours**.)
- **With the same level of** allows you to easily locate points related to the selected points.

  The main use of this would be to locate points with the same level of a chosen factor (e.g., if you are exploring survey data, you may wish to identify observations in the same cluster), or you may want to retain points selected over multiple graphs (e.g., in the Gap Minder data set, included in the `Data` folder, has several observations of countries over multiple years. In this case you can label certain coutries and track them over time).

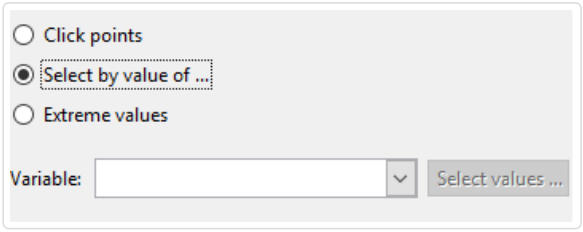
  Note: when you do this, the point you clicked will be highlighted for easier reference.

How do you want to select points?



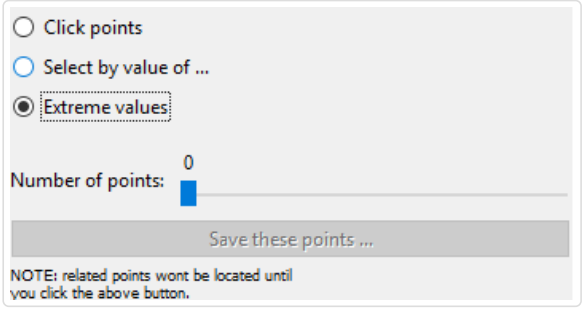
- **Clicking with the mouse**: After clicking "Click to Locate ...", you can then click a point on the plot. iNZight will then label it using the options you defined above. You can select multiple points by clicking the button again and locating new points without losing the current selection.

  Note: due to the way the software works, you may see a "Busy" cursor after you click the "Click to Locate ..." button. This will not go away until you click a point, so go ahead and click points ignoring the cursor.



- **Select by value of ...**: this will allow you to select a variable to use to select points.

  If the variable is a factor (or a numeric with less than 20 unique values) a slider will appear allowing you so quickly select the levels you wish to label. If the variable is numeric, or you want to select multiple levels, you can click the "Select levels ..." button to do so.



- **Extreme values**: This will use a very simple algorithm that will select the points "farthest away" from the bulk of the data.

  This will display a slider, allowing you to select more or less points as desired. Once you have selected the points, you can optionally save the selection (e.g., if you want to track the same observations over multiple plots) you can click the "Save these points ..." button. From there you can use the "With the same level of" option to select related points, etc.

The statistic used is **Mahalanobis' distance**.

Previous: Plot Toolbar Buttons

Plot Options

Next: Inference Information

Add Inference Information

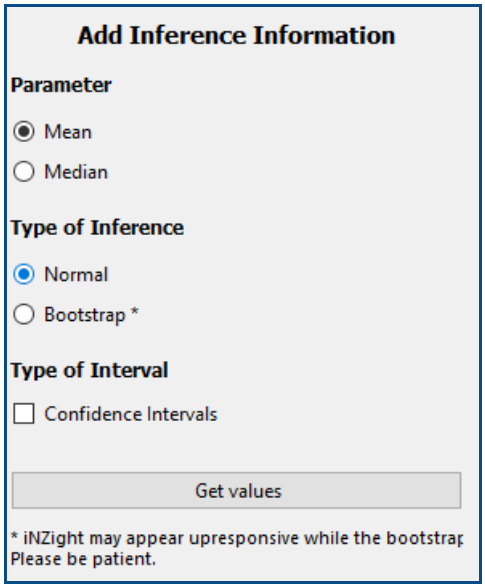
iNZight provides users with an easy interface for adding inference information to plots, using "normal theory" or bootstrap methods. The inference options available again depend on the type of plot drawn.

Scatter Plot Inference

If a scatter plot is drawn, you must first add trend lines before you can obtain inference information. This is done using the **Add to Plot** button, selecting **Add trend line** and selecting one or more trends (this includes the smoother).

Having added a trend line or lines, the inference information will generate 30 bootstrap trend lines and display these on the graph, giving users an idea of the precision of the estimated curve.

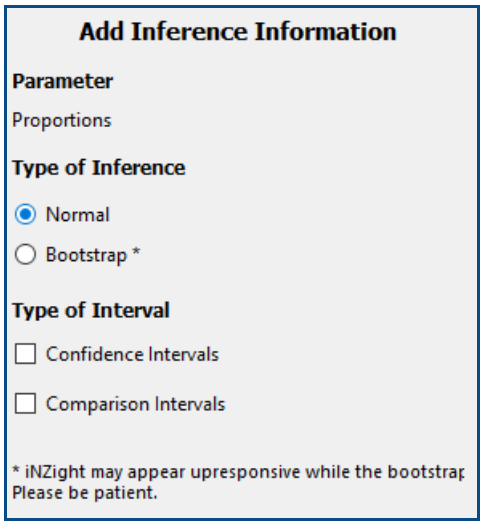
Dot Plot Inference



Dot plots, by default, provide a box plot which provides an estimate of the Median, as well as the 25% and 75% quantiles. Users are then presented with three options:

- **Parameter**: the parameter for which inference is obtained
    - **Mean**: this will generate inference of the mean of the data
    - **Median**: this will generate inference of the median of the data
- **Type of Inference**: Depending on the parameter, users can select the type of inference used to obtain the intervals displayed on the graph
    - **Normal**: this uses standard normal theory to approximate a confidence interval
    - **Year 12**: for **medians**, the year 12 (NCEA L3) method of computing an interval is used
    - **Bootstrap**: this method uses bootstrap sampling to obtain confidence or comparison intervals of the chosen parameter
- **Type of Interval**: the type of interval displayed
    - **Confidence Interval**: this displays a 95% confidence interval
    - **Comparison Interval**: if a numeric and categorical variable have been plotted, comparison intervals can be added to allow visual comparison of the mean or median of the numeric variable between different levels of the categorical variable.

Bar Plot Inference



The inference information for bar plots is computed for the estimated proportions, using normal theory or bootstrap techniques.

Previous: Add to Plot

Plot Options

iNZight User Guides: Advanced Modules

The Advanced modules here are for special forms of analysis for specialized types of data.

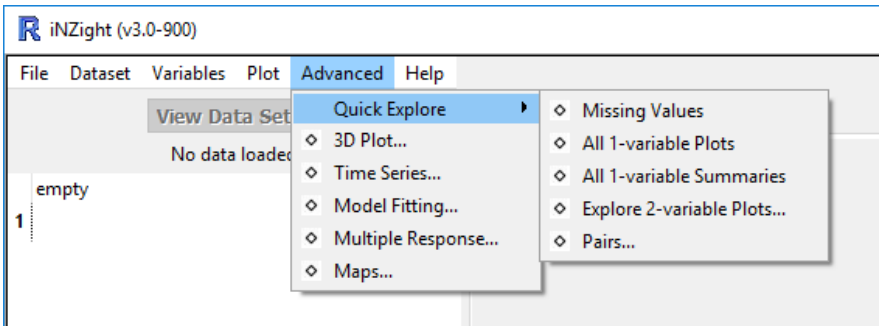
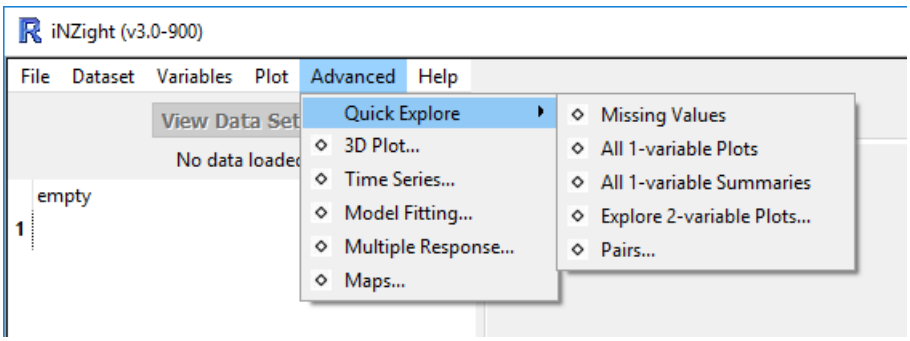


Table of Contents

Quick Explore

This menu provides several exploratory analysis tools for quick familiarisation to a new data set.



Missing Values

Plots and summary tables of the extent to which variables and combinations of variables have missing values.

All 1-variable Plots

Steps through and plots every (single) variable in the data set.

All 1-variable Summaries

Summaries for every (single) variable in the data set.

Explore 2-variable Plots

Takes a user-selected variable and then steps through plotting every other variable in the data set against it.

Pairs

Takes a set of user-specified variables and plots every variable against every other variable as a matrix of plots. There is an option to colour using another variable. (Uses **R**'s `gpairs` package).

Interactive 3D Plot

This module provides users with the ability to plot three **numeric** variables simultaneously on a 3-dimensional axis. A fourth categorical variable can be used to colour the points. The image can be rotated and spun, allowing user to explore relationships between variables.

Provides access to code from **John Fox's *R Commander*.**

Previous: Quick Explore

Advanced Modules

Next: Time Series

Time Series Analysis

Graphics for time series aimed at seasonal series.

Includes time series plots with smoothed trends, plots displaying seasonal decomposition, forecasting using Holt-Winters, and plots for comparing a set of series.

Caters for constant additive or multiplicative seasonal effects.

Specifying the Time Structure

Users can supply time information to iNZight in one of two ways: either by specifying an appropriately formatted variable, or by manually telling iNZight about the structure.

Supported Time Variables

iNZight's Time Series module can currently only read specially formatted variables to determine the time structure of a data set, and supports observations made over periods of a year (daily, weekly, monthly, quarterly, yearly observations), daily observations made over weeks, and hourly data made over days.

The following table shows the formula for these time variables (code letters in parentheses {} are optional)

Period	Frequency	Code	Examples	Notes
Yearly	Annual	{Y}*yyyy*	1990, Y1990	
	Quarterly	{Y}*yyyy*Q*qq*	1990Q01, Y1992Q01	*qq* can be 01-04
	Monthly	{Y}*yyyy*M*mm*	1990M01, Y1990M01	*mm* can be 01-12
	Weekly (7-days per week)*	{Y}*yyyy*W*ww*	1990W01, Y1990W02	*ww* can be 01-52
	Daily	{Y}*yyyy*D*ddd*	1990D01, 1990D127, Y1990D001	*dd* can be 001-366
Weekly	Daily	W*ww*D*dd*	W01D01, W125D07	*dd* can be 01-07
Daily	Hourly	D*dd*H*hh*	D01H01, D184H23	*hh* can be 01-24

* if your data has a period of 5 days per week, then you'll need to specify the time manually (see below).

Manual Time Specification

If your data doesn't fit one of the above structures, or you don't want to manually create the variable, you can easily specify the structure of your time series data by choosing "Provide Time Information".

- Use the table above to determine what the **period** and **frequency** of your data should be, and then choose the appropriate options from the drop downs.
- You can specify you own ("custom") frequency if it is not one of the common ones listed above; for example, if you have daily data over a period of **work weeks** (Monday – Friday), set the frequency to 5.

Additional Documentation

We have very little documentation ourselves, however **CensusAtSchool NZ** has some **resources related to time series**.

Model Fitting

A modelling module for (multiple) linear regression, logistic regression, and Poisson regression.

It also caters for data from complex survey sampling.

Extensive diagnostic and other plotting facilities with many novel features.

Multiple Response

A module that provides the ability for analysing data with multiple (binary) response options. Think of this as a checkbox answer, as opposed to a radiobox:

Standard factor variable
What's your favourite fruit?
○ Apple ○ Banana ○ Pear ○ Orange

Multiple Response variable
What fruit do you like?
☐ Apple ☐ Banana ☐ Pear ☐ Orange

The first only lets you provide one option, so the variable only has one column in the dataset:

person.id	fav_fruit
1	pear
2	apple
3	orange
4	apple

The second, however, needs a column for each possible answer:

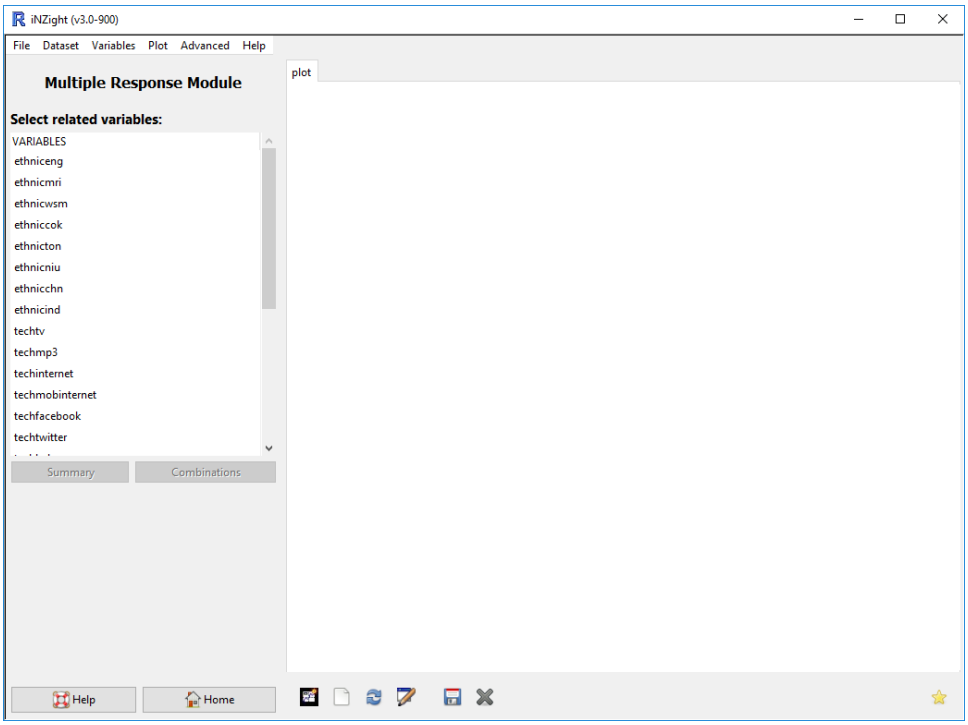
person.id	fruit_apple	fruit_banana	fruit_pear	fruit_orange
1	1	0	1	0
2	1	0	1	1
3	1	1	1	1
4	1	0	0	0

Analysing these are going to require two difference approaches, and that's where the Multiple Response module comes in!

Quick note on coding: in the table above, we've used 1's and 0's to code whether or not someone checked the box. This is the easiest way, however you can also provide "yes" (=1) or "no" (=0) if you prefer.

Getting Started

First, you'll need to load an appropriate dataset (i.e., one that has data coded like the table above). For this example, we will be using the **Census at School 5000** dataset, which you can load by going to **File** > **Example data**, select **Multiple response** as the module, and **census.at.school.5000** for the data. Once you've loaded this, you can go to **Advanced** menu and click **Multiple Response**, which will load the module.

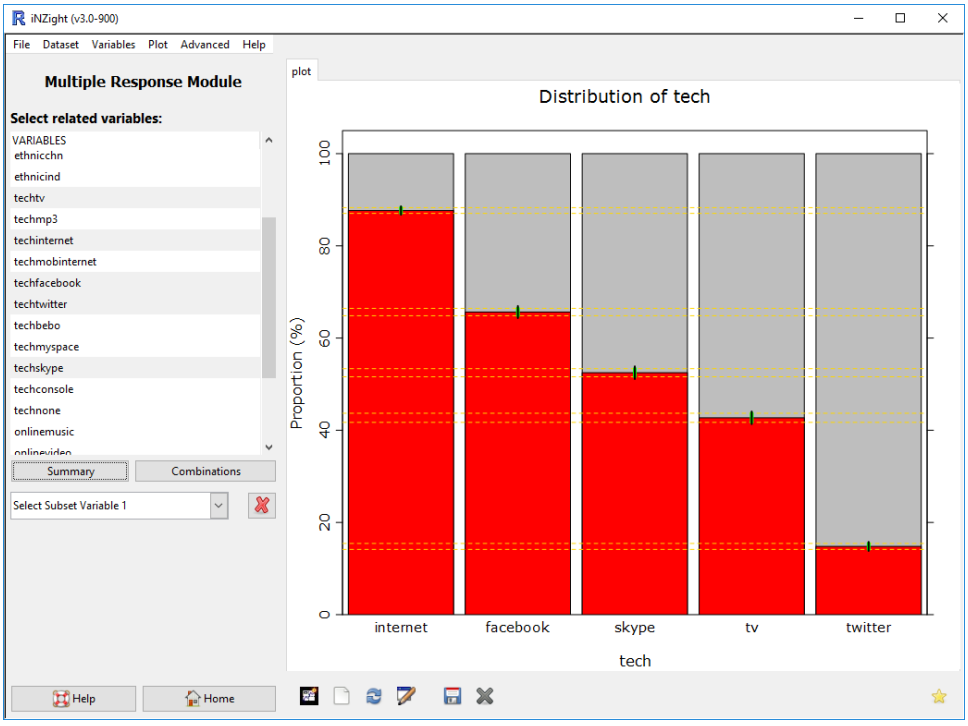


The **Census at School 5000** dataset contains several variables, each with multiple responses. Here, we will investigate **tech**: each student was asked to select all technologies they have used.

Generating Graphs

Once you've loading the Multiple Response window, you'll be able to select variables to compare. In our example, we will use the variable **tech**, and select the respones **tv**, **internet**, **facebook**, **twitter**, and **skype**.

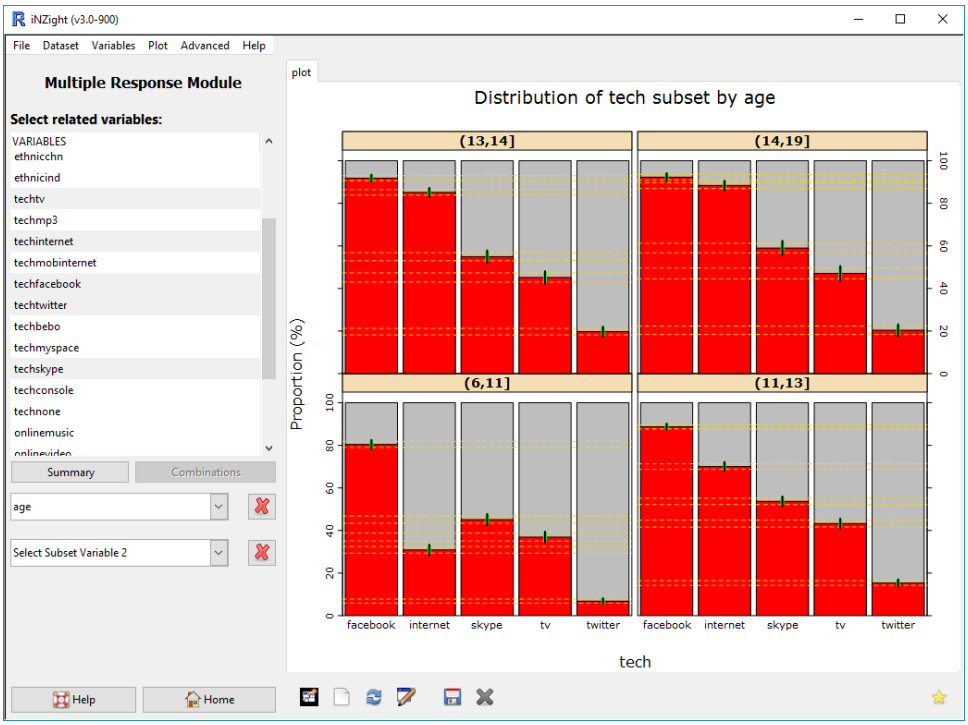
HINT: To select multiple variables, hold the `CTRL` *key (or Command on Mac).*



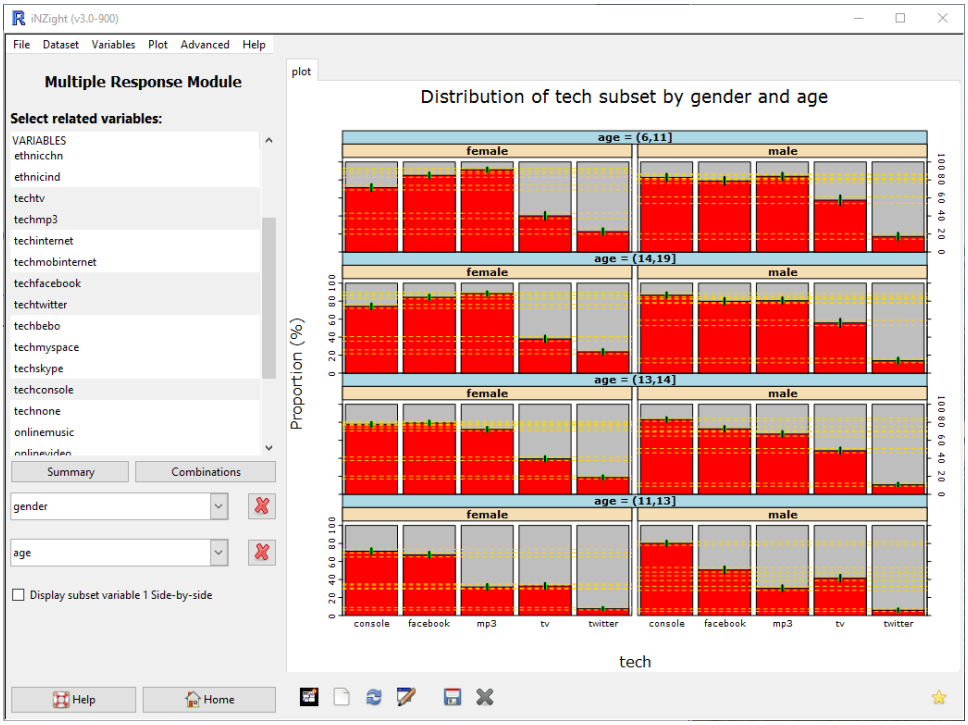
The Summary and Combinations buttons will give you summary tables of the data, and a "visual" graph of the combinations occurring in the dataset.

Sub-setting the Graph

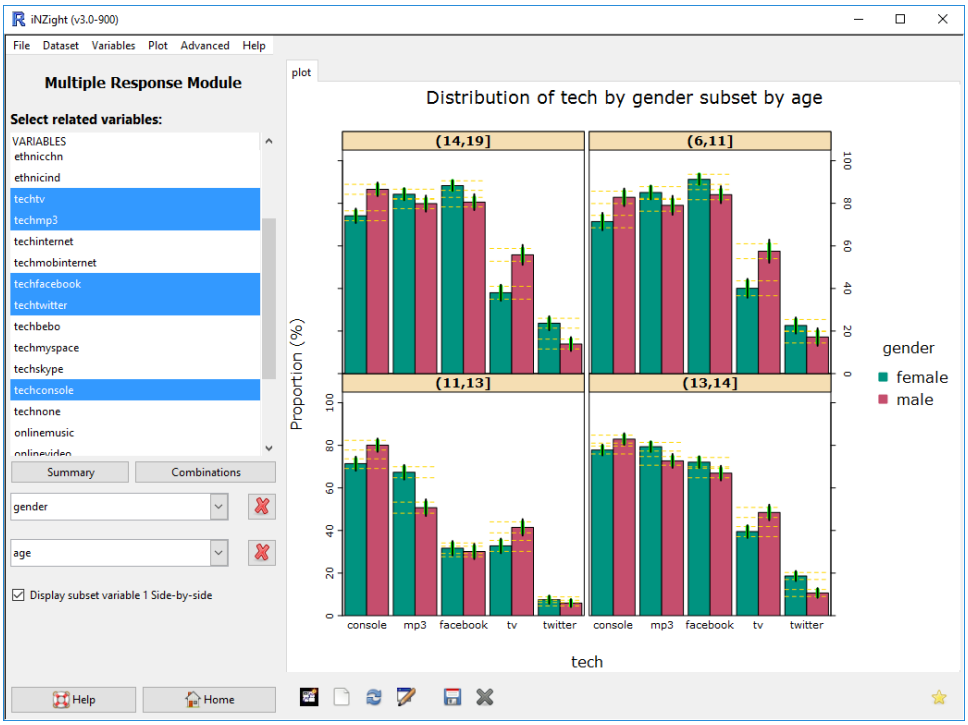
As with the main iNZight module, you can split the graph into several **subsets** to better explore trends. In the example below, we have selected **age** as the sub-setting variable. Because age is a numeric, iNZight has divided it into four groups.



To delve further into trends, you can use two sub-setting variables. First, we'll pick a new set of responses: **tv**, **mp3**, **facebook**, **twitter**, and **console**. The first sub-setting variable is **gender**, and the second is **age**.



If we want to more easily compare the first sub-setting variable, we can check the box **Display subset variable 1 side-by-side**:



Previous: Model Fitting

Advanced Modules

Next: Maps

Maps

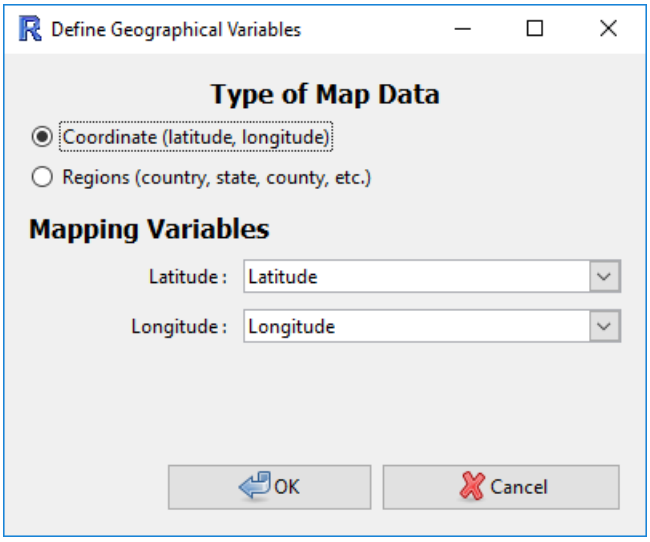
If you have never used the Maps module

Plotting data on maps is a specialised area that is quite different from normal graphing. **The article here** tells you about the sorts of data that we can present using maps and ways in which we can use maps in constructing data visualisations.

The Maps module is not part of the initial installation of iNZight. It downloads and installs automatically the first time you try to launch it by going to the **Advanced > Maps**

Selecting data type

The iNZight Maps module can handle two types of data, ***Coordinate*** data and ***Regional*** data. The first step is to tell the program whether you have *Coordinate data* or *Regional data*.



The program behaves differently depending on this choice. The documentation deals first with Coordinate data and then with Regional data.

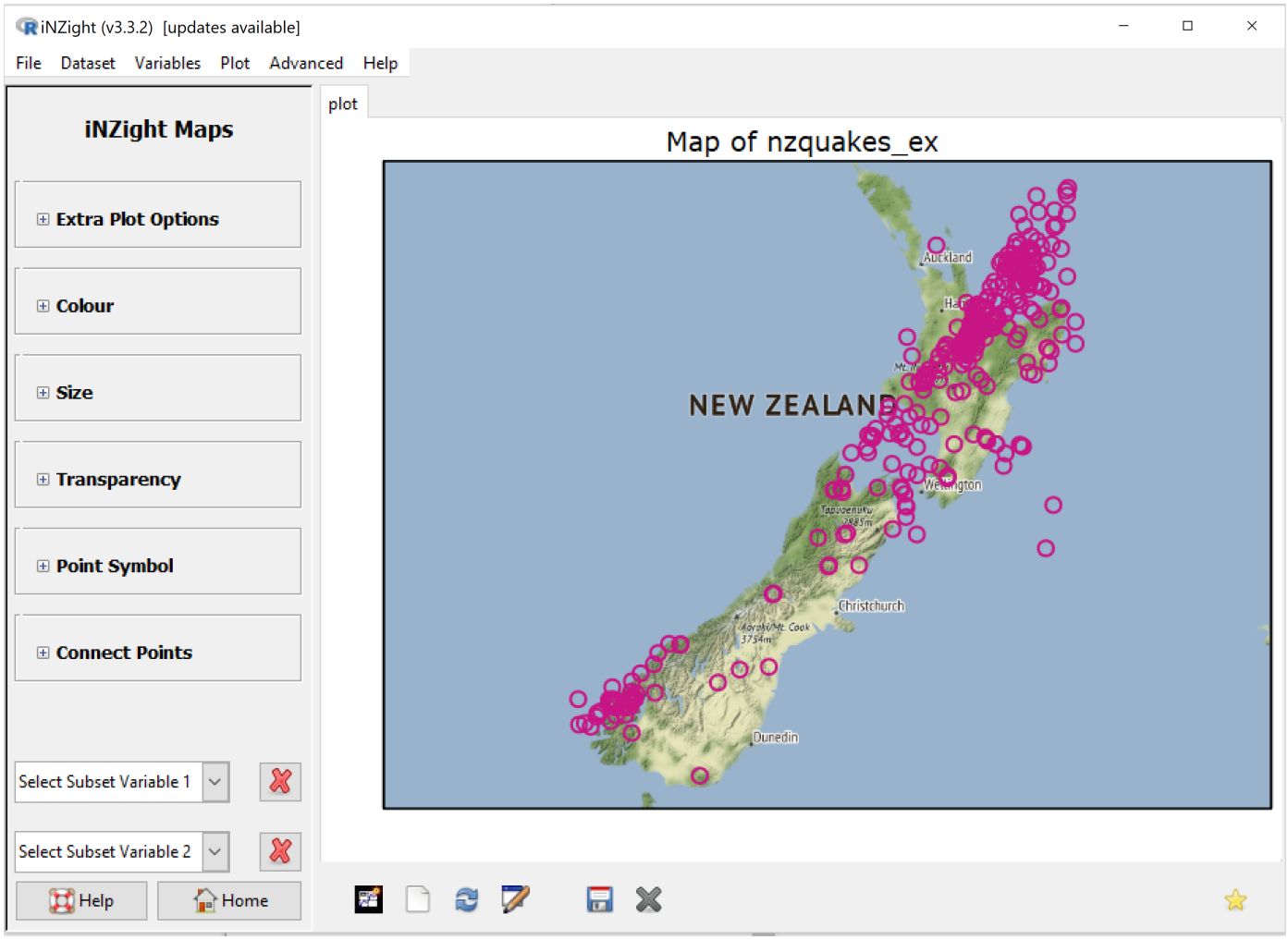
- ***Jump to ...*** **Coordinate data** | **Regional data**

The **nzquakes** data obtained available from **File** > **Example data** and selecting the **Maps** module is an example of *coordinate data*. The **NSSATS** data also available there, is an example of *regional data* (data about US states).

Geographical Co-ordinate Data

iNZight will try to guess the **Latitude** and **Longitude** variables automatically, but you will have to select them manually if they have nonstandard names.

Once you've selected those, ***click*** **OK** and iNZight will now plot your data on a map (it may take a little while for the underlying map to download)



[***This only works if you are connected to the internet*** because iNZight Maps ***downloads*** appropriate **Stamen map** tiles (via ggmap) to form the background of the plot]

Plot appearance can be changed and information about other variables added to the plot using the following controls.

- Click on the **+ *buttons*** to expand the option panels

Extra Plot Options:

iNZight Maps ***downloads*** appropriate Stamen map tiles (via ggmap) to make the background of the plot

Map-type: options are ***terrain*** (*default* - coloured terrain map with some features named), ***terrain-background*** (no added names), ***toner-lite*** (white map on grey background) and ***toner*** (white map on black background).

Overall point size (*Slider*)**:** scale everything drawn on top of the plot/map to a chosen multiple of their current sizes (default =1)

Colour:

Either choose a ...

Point colour: select a single colour to apply to all of the points

Or select the names of a ...

Colour by *variable* to differentially colour points according the values of the chosen variable.

Additional options *which appear once a* ***Colour-by*** *variable has been selected:*

- **Palette** for selecting a colour palette
- **Reverse palette** (*check-box*)**:** Reverse the way the values of the colour-by variable are mapped to colours (smallest-to-biggest/biggest-to-smallest)
- **Use percentiles** (*check-box*)**:** Apply the colour range according to the percentiles of the colour-by variable instead of its raw values.
- **Cycle levels** (*categorical colour-by variable*)**:** Clicking the arrow icons results in just the points from one category at a time being highlighted, with the others dimmed
- **Cycle quantiles** (*numeric colour-by variable*)**:** Behaves in the same way with just the points in a quantile being highlighted. By default just 4 quantiles are used (useful if the colour-by variable is highly skewed), but ...

    - **#quantiles** allows you to change 4 to some other number of quantiles

- **Back cycle *icon*** results in all of the points being highlighted again

Size:

Overall (*slider*)**:** Multiply the current size of each point by value chosen (default value 1)

Size by: Select a variable to differentially size the points by (i.e., size according to the values of the chosen variable)

***Additional option** which appears once a **Size-by** variable has been selected*:

- **Resize method:** make the area of the point either ***proportional*** to the value of the size-by variable, or ***emphasize*** (map the smallest values to a hard-coded small point-size and the largest to a large size – sometimes useful when the ratio largest/smallest is not very big)

Transparency/Opacity:

Overall (*slider*)**:** Adjust the transparency levels of all of the points uniformly (default value 0 = no change; maximum makes all points completely transparent)

Opacify by: (Opacity is the opposite of transparency) Select a variable to differentially set the opacity the points by (high values map to most visible, low to most transparent)

***Additional option** which appears once a **Opacify-by** variable has been selected*:

- **Reverse opacification** (*check-box*)**:** Reverse the way the values of the opacify-by variable are mapped to opacity/transparency levels

Point Symbol:

Symbol: Select from *circle*, *square*, *diamond*, *triangle* and *inverted triangle*

Filled in symbols (*check-box*)**:** Click to fill in all symbols

Symbol by: Can be done according to a categorical variable which has 5 or fewer categories

- **Symbol line width:** Line-width multiplier

Connect Points:

Connect points with lines (*check-box*)**:** Check the box to draw lines between points. Connected in the order they appear in the data file.
***Additional options** which then appear*:
● **Line colour** (*selection box*); ● **Line width:** (*slider*)

Sub-setting:

Works the same way as in the ***main part of iNZight***, you can generate subsets by selecting a variable from either one or both of the subset menus.

Subset selection ***sliders*** and ***play buttons*** *appear* once a subset-by variable has been selected.

Interactive Coordinate Maps

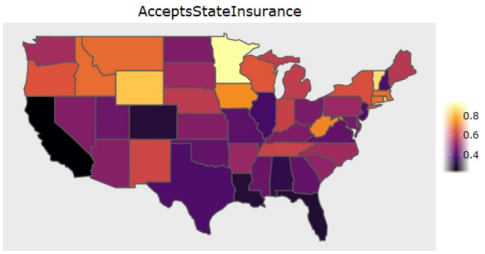
When you click on the ***Interactive plot icon*** (immediately above), iNZight makes a temporary interactive html file and opens it using your default browser.

- Interactive versions of the maps are ***not available when there is more than one map*** in the plotting window

To save a permanent file, click on the ***Save Plot icon***, or go **Plot > Save Plot**.
Choose the **interactive HTML** as your file type.

These files work independently of iNZight

Data on Geographical Regions

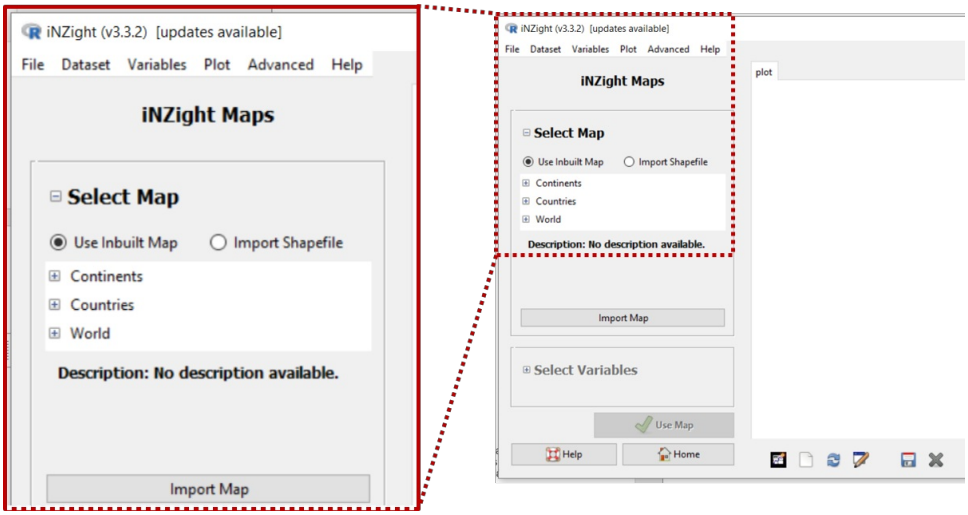


In many cases, you might have data with regional information (where "region" can be countries, states, provinces, electorates, etc.).

iNZight allows you to display this information on a map of outlines, with each region coloured according to the value of a chosen variable. (Some alternative representations are also catered for.)

- *Jump to ...* **Selecting a map** | **Plotting data on the map** | **Multiple observations per region** | **Interactive maps**

.... Selecting a Map



Select Map

Choose either **Use Inbuilt Map** (*default*) to use one of the maps already in iNZight
(choose from a selection of maps of *Continents*, several *Countries*, and *World* maps)
or **Import Shapefile** to start up a dialog for reading a shape file stored on your computer

Double clicking on a map name in the **Use Inbuilt Map** section will plot a preview of the map in the plotting area and display a brief description of the map.

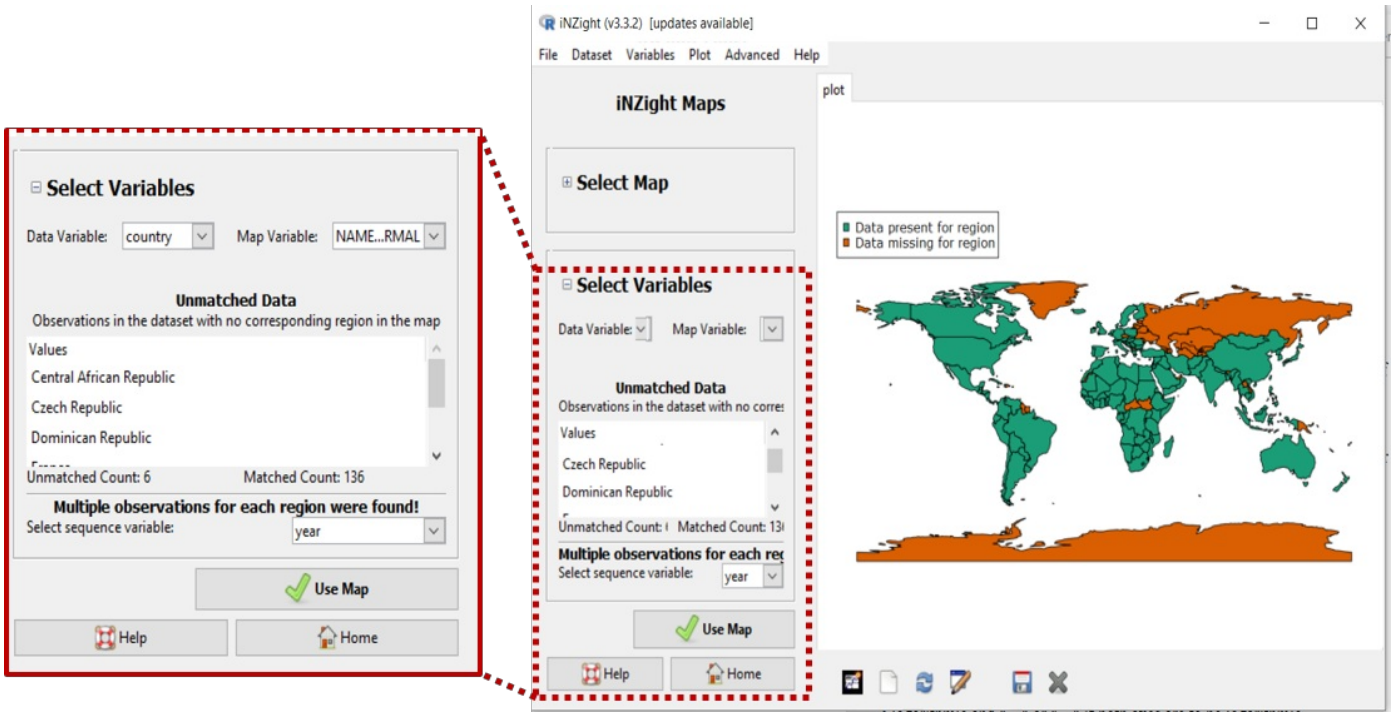
Then click ... **Import Map**

Select Variables

When a map has been imported, iNZight will also try to find a variable in your data file which best matches one of the region-naming variables on the map. It will display the names it has found beside **Data variable** and **Map variable**. Selection boxes allow you to select replacement variables for these two roles.

- ***Selection boxes*** allow you to ***change the variables*** used for each of these two roles.

It will also draw the map. Regions for which *data is present* in the data set are shaded *green*.
Any Regions which have *no data* are shaded *orange*.



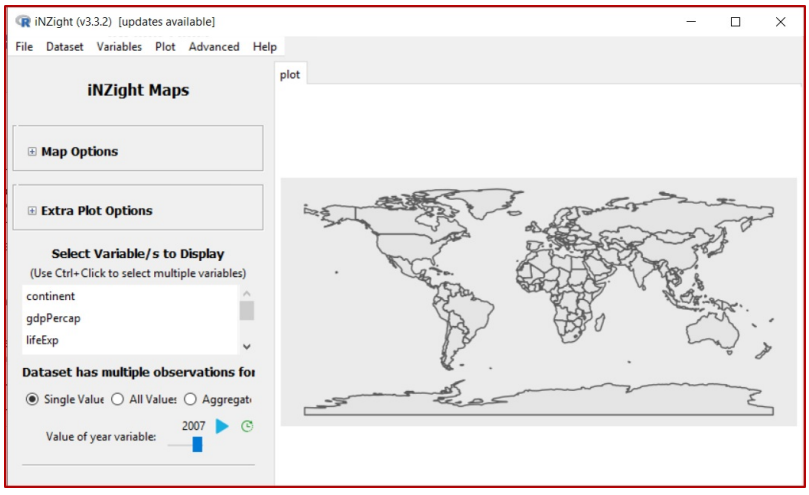
Unmatched Data:
The names of any Regions in the data that are not represented in the map will be listed here once the map has been imported. You can decide to use the map anyway or exit maps and go back to your data to do something about the non-matching names.

Multiple observations for each region were found: (*only appears when this is true*)
This often happens when, for example, we have observations for a region for several different years with different years corresponding to different rows in the data set.
The module will then try to find a variable that "explains" the multiple observations. We call this a "*sequence variable*".

- **Select sequence variable:** If an “explaining variable” is found its name will be displayed in the selection box. You have the freedom to change it

Use Map (*Button*)**: Click** to tell the program "***I want to use this map***"

- The program will then replace the map-selection command panel with a new command panel for plotting. An unshaded map with the appropriate boundaries will appear (as follows)



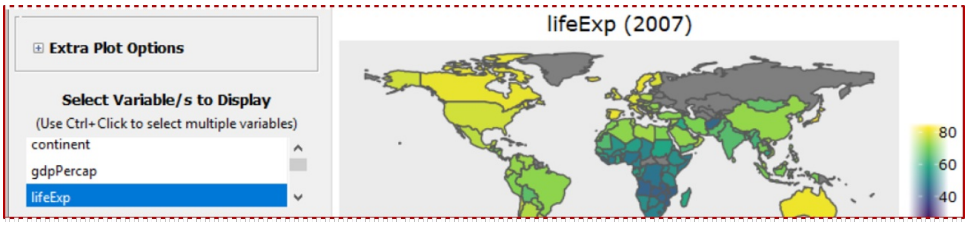
.... Plotting data on the Map

If you are using this module for the ***first time*** it is best to ***ignore*** the **Map Options** and **Extra Plot Options** paragraphs initially and see how the plotting of variables works. So that is what we are going to discuss next.

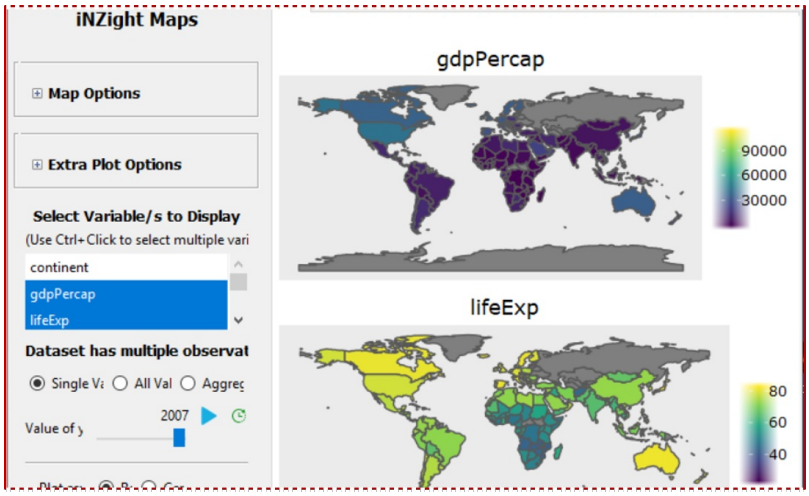
- *Jump to* ... **Map Options** | **Extra Plot Options**

Select Variable/s to display

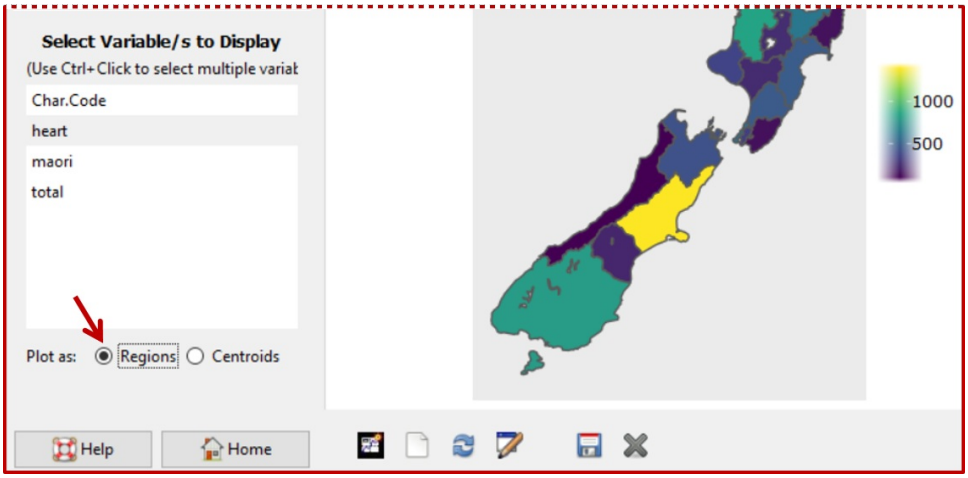
Click on the name of the desired variable and it will display on the map



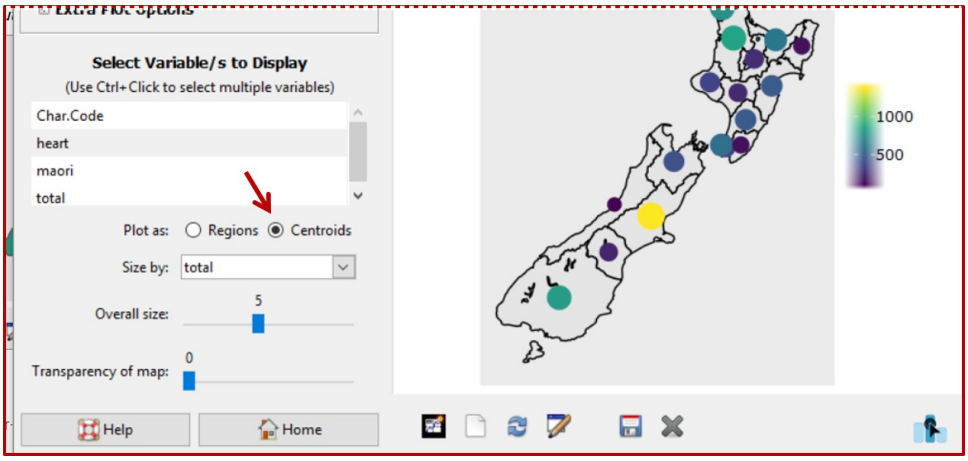
If you ***select several variables*** you will get a map plot for each of those variables ...



Regions *radio button* gives ***shaded maps*** ...



Centroids *radio button* gives ...

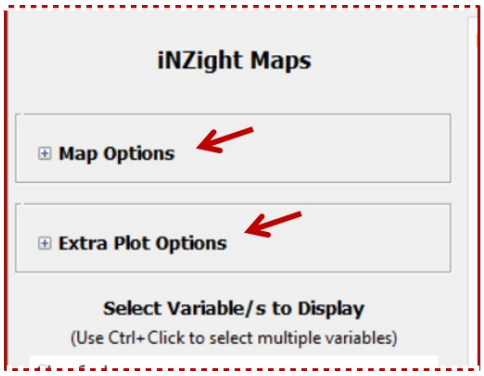


Additional options *that become available* ***when Centroids*** *is turned on*:

Size by: (*typically use a relevant population-size variable*)

Overall size (*slider*)**:** makes the disks bigger or smaller uniformly

Transparency of map (*slider*): transparency of map outlines



Map Options

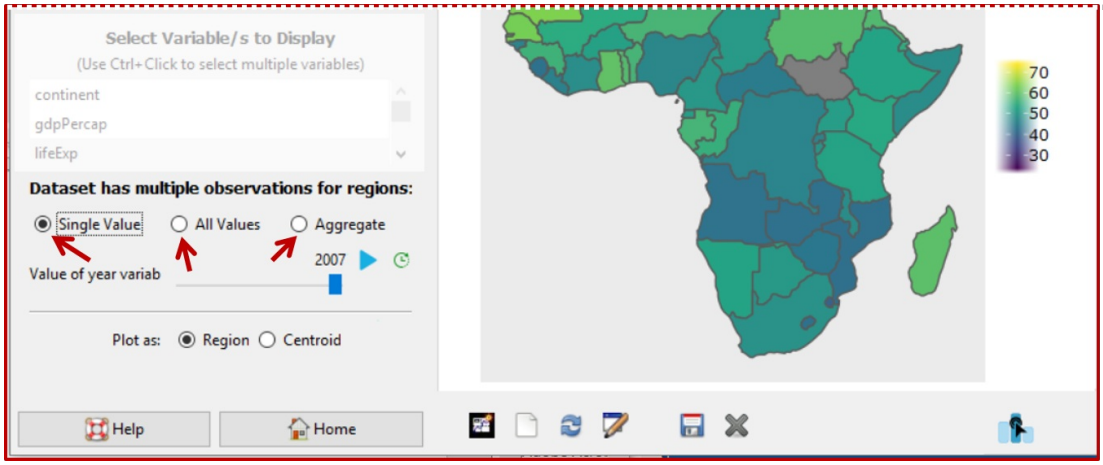
Current Map: Opportunity to change your mind and select a different map
Regional selection boxes: use to remove regions from the plotted map
Projection (*dropdown list*)**:** Opportunity to change the map projection (default is the projection suggested by the shapefile)

Extra Plot Options:

Plot Title: change plot title
Palette: change colour palette
***Check boxes*: Dark** (makes sea go dark), **Grid Lines** (use to add), **Axis labels** (use to add)
Map Scales: Choices for *independent scales/same scale* to be used across a set of plots
Font Size (*slider*): **Region Labels** (*check box*)**:** add labels
***Additional options** that become **available when Region Labels** are turned on* :
Selection box for selecting ***variable*** to use ***for labelling***
Region label font size: (*slider*)

Dataset has multiple observations per region

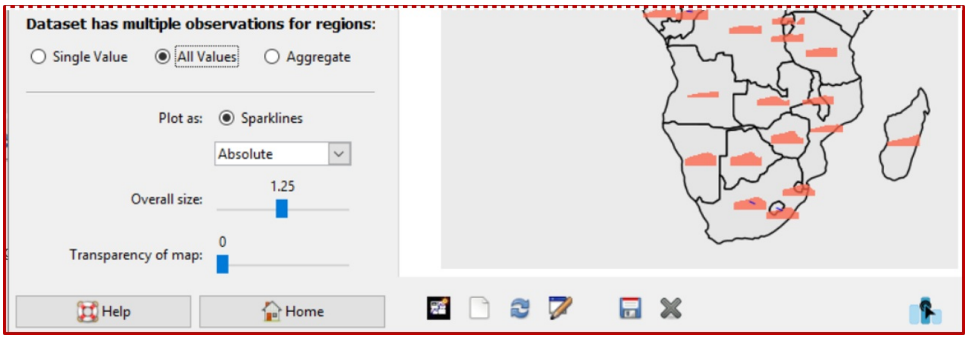
(*only appears when this is true*)



- **Single value:** display just the last value
  ***Additional options** that become available **when** **Single Value** is chosen*:
  Value of ... variable (*slider*) **:** choose value of the explaining variable (e.g. year)
  Play button and ***playback timer button*:** controls playing through the values/images

- **Aggregate:** display an aggregate of the values (e.g. mean, median)

- **All Values:** display mini time-series glyph ("***sparklines***") for each region
  ***Additional options** that become available **when** **All Values** is chosen*:
  Sparklines can be plotted for ***Absolute*** (the raw numbers), ***Relative*** (ratio to the first value), ***Percent Change*** (between successive observations)
  Overall size (*slider*) **:** controls size of the sparkline glyphs
  Transparency of map (*slider*)**:** transparency of map outlines



Interactive Maps

When you click on the ***Interactive plot icon*** (immediately above), iNZight makes a temporary interactive html file and opens it using your default browser.

- Interactive versions of the maps are ***not available when there is more than one map*** in the plotting window

To save a permanent file, click on the ***Save Plot icon***, or go **Plot > Save Plot**.
Choose the **interactive HTML** as your file type.

These files work independently of iNZight

Shape files and importing shape files

Shapefiles are a method of storing geographical data for use in GIS (geographical information systems) or similar software. Despite the singular name, shapefiles are actually a collection of files that each correspond to a particular aspect of the data. Three files are mandatory:

- The geometry of the data (polygons in our case) in a **.shp** file
- An index of the geometries contained in the .shp file to allow software to know where each geometry is located within the file. This is contained in a **.shx** file.
- Attributes for each geometry contained in the **.shp** file. These attributes can be any data and function similarly to a data frame where each row corresponds to a single geometry. These are stored in a **.dbf** file and can be inspected with Microsoft Excel.

Most shapefiles will have additional files containing other information such as map projections or indexes of the spatial distance between geometries. Shapefiles are typically stored in separate directories for each map, with all the files that make up the shapefile contained within a single directory. Shapefiles are generally distributed on the internet as archive files, e.g. ZIP files. Currently these have to be decompressed to use in iNZight.

- In the **Import Shape file** dialog, open the **.shp** file in the relevant shapefile folder

Other shapefile-information file formats that can be used in iNZight are **GeoJSON** and **TopoJSON files**. These are a single file that stores much the same information as a shapefile by storing a geometry and set of properties for each feature in the dataset. GeoJSON can be used in most GIS software and is a standard format for web-based mapping technologies such as Leaflet. TopoJSON is a closely related format that includes topology and typically has file size that is smaller than GeoJSON as TopoJSON only stores one copy of any borders that are shared by two countries.

- In the **Import Shape file** dialog, just open the relevant **.GeoJSON** or **.TopoJSON** file

****Coming*** (Daniels's notes)**:**

- Add ability to import zipped shape files as is
- A newer, faster method of map simplification
- A more integrated map selection interface where we download less common shapefiles on-demand, but still list them in a tree diagram
- As an additional note, my current method of simplifying our “built-in” maps is to use a webpage that uses this method. Once I put it into a script, I can send it to you and we won’t need to jump between different software to add a map to your server (or perhaps more importantly, not rely on other services)

Previous: Multiple Response

Advanced Modules

Next: About Maps ›

About Presenting Data on Maps

This article gives an overview of plotting data on maps. It is not software documentation. (**The software documentation is here**)

Plotting data on maps is a specialised area that is quite different from normal graphing. This article describes the sorts of data we can present using maps and ways in which we can use maps in constructing data visualisations to help ourselves and others better understand that data.

We start by outlining the types of data that are appropriate for presentation on maps and then go on to show ways of visualizing data using maps; addressing important issues about using maps for data visualization as they arise.

Everything you see illustrated in this article is very fast and easy to obtain using iNZight's Maps module

The types of data most commonly plotted on maps

The two types of data most commonly plotted on maps are ***location*** (or geographical coordinate) data and ***regional*** data.

- **Location (or Graphical Coordinate) data:** This is data on things that have happened at particular *geographical* ***locations***. The most widely understood way of specifying a location is by its ***Latitude*** and ***Longitude***, though there other location coordinate-systems. Additional variables give information about what happened there,
  - **Example:** the data below on earthquakes in New Zealand in the year 2000 where we have information about where the epicentre of the quake was (latitude and longitude), how deep underground it was, how strong it was, when it occurred, and several other variables.

Order	Latitude	Longitude	Depth	Felt	Magnitude	RMS	NorMidSth	Month	Day	Hour	Minute
1	-39.14347	174.8942	-218	N	4.289	0.19	North	1	3	11	53
2	-37.70441	177.2514	-65	Y	4.454	0.21	North	1	4	18	44
3	-40.25043	173.57	-196	N	4.035	0.247	North	1	6	19	19
4	-38.14048	176.3235	-158	N	4.479	0.205	North	1	7	12	30
5	-36.96741	177.0199	-212	N	4.049	0.183	North	1	7	15	30

- **Regional data:** This is (region-level) data on different geographical regions. By regions we mean things like countries of the world; or states/counties/electoral districts within a country. The associated variables usually give summary measures for each region,
  - **Example:** the table extract below shows data on different countries of the world which includes: the year the data relates to, average life expectancy, population size, and GDP per capita.

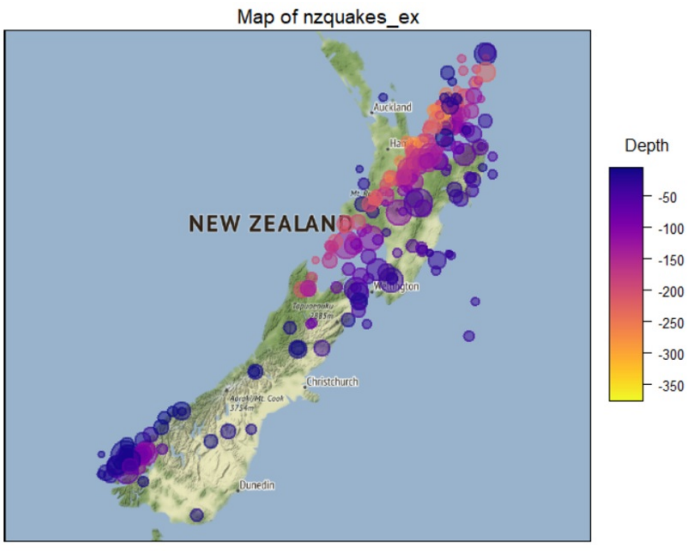
country	continent	year	lifeExp	pop	gdpPercap
Algeria	Africa	2007	72.301	33333216	6223.367
Angola	Africa	2007	42.731	12420476	4797.231
Argentina	Americas	2007	75.32	40301927	12779.38
Australia	Oceania	2007	81.235	20434176	34435.37
Austria	Europe	2007	79.829	8199783	36126.49
Bahrain	Asia	2007	75.635	708573	29796.05

(*In this data,* continent *could also be used as a region variable if our data was recorded at the level of continent*)

Statistical exploration of data using maps is typically concerned with ***finding patterns in the data that are related to geography,*** leveraging off what we can see about what is close to what (or far from what), and the reminders the maps give us of our own general knowledge about different locations. We are particularly looking for patterns that make some sort of sense (or are interesting) geographically, and any notable exceptions to those patterns.

Plotting Location data

With location data we start by plotting the locations (latitudes and longitudes) on the map using some form of plotting symbol (e.g. a circle). But doing that that just shows us the locations we have data for. It does not communicate any of the information we have about what happened at each of those locations. So we need ways of putting the additional information from some of these other variables onto the map as well.



(*We have sized by the intensity of the earthquake and coloured by its depth*)

Adding more information than just locations

(*for static, dynamic and interactive maps*)
Relate the values of different variables to some of the following:

- ***Size*****:** the size of the plotting symbol
- ***Colour*****:** the colour of the plotting symbol
- ***Shape*****:** the shape of the plotting symbol
- ***Opacity*****:** the opacity vs. transparency of the plotting symbol
- ***Labels*****:** Put labels by the points (names, numbers)

Also,

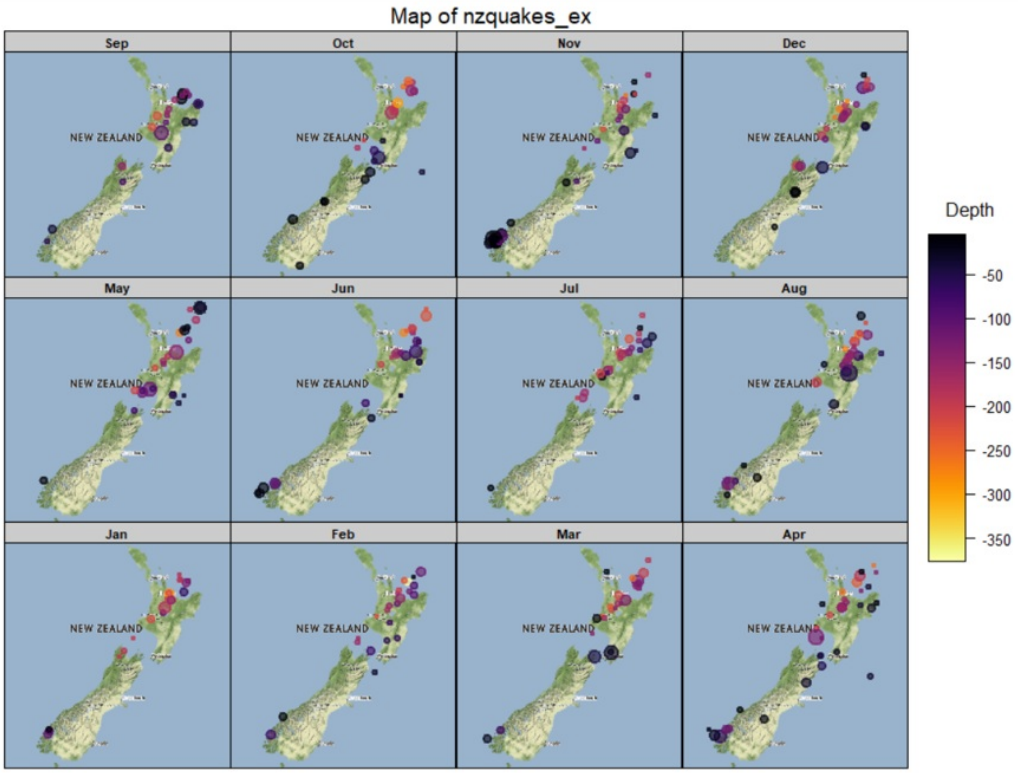
- ***Connect points by lines*****:** if the data is in time order this conveys information on the order in which things happened (and, e.g., trace the migration path of a whale)
- ***Have the points appear sequentially in time order*****:** another way of conveying the sequence of events (*not implemented in iNZight yet*)

(*for interactive maps*) (- *jump to* **interactive maps**)

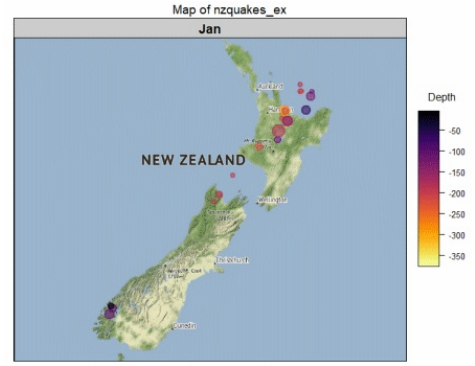
- ***Hover over*****:** Display the values of variables when the mouse hovers over a point
- ***Select/Brush*****:** When points on the map are selected in some way, initiate some sort of new display. This make be as simple as showing labels and variable values, or we could throw up some new graph, or highlight related features in another graph

Subsetting/Faceting

- ***Array of maps*****:** let's you see a lot of maps at once and compare them
- ***Playing through a sequence of maps*****:** Can be good for looking at changes (e.g. over time)



(*Here we have a different map for every month of the year*)



(*Small animated gif to show the effect of playing through subsets*(*here, playing through the month plots*))

Interactive Map plots

Example

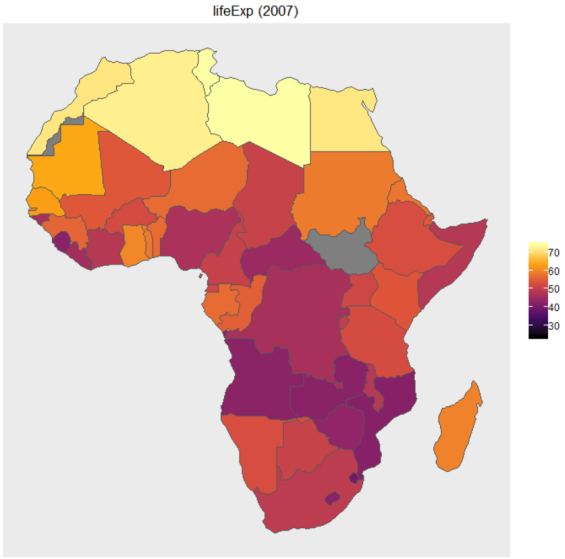
Here is an interactive version of the February plot. This one responds to hovering over points, and selecting a set of points by left-clicking and dragging. Which variables get displayed when hovering is controlled by the "Variables to display" dialog a the top of the frame. When a set of points is selected by dragging, their data values appear in the Table. Clicking on "Table" at the top of the frame toggles between removing the Table and bringing it back again. (*Click the graphic's help page -- small "?" at top right.*)

Contour plotting

(cf. the contours of equal atmospheric pressure ("isobars") on a weather map, or of equal altitude on a topographical map -- *Still to be written- not implemented in iNZight yet*)

Plotting Regional data

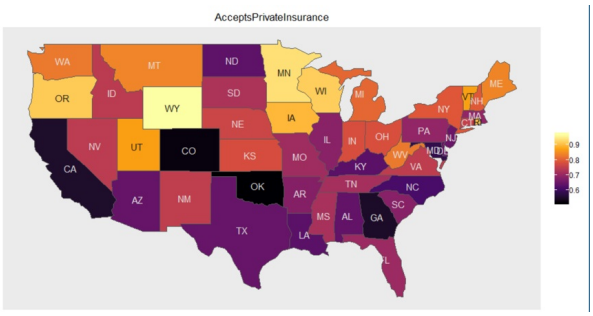
Typically people colour regions of a map according to the value of a (single) variable. Such maps have historically been called ***choropleth maps***. If the colouring has been done according to the values of a continuous variable the result is sometimes called a *heat map*. (Different shading patterns are sometimes used as an alternative to using different colours.)



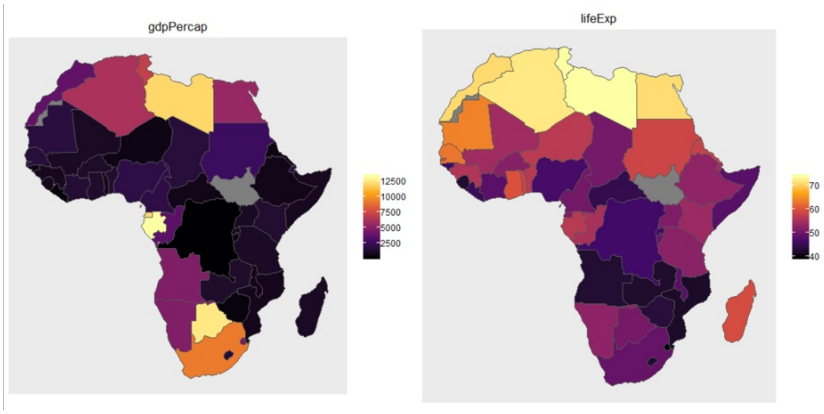
(*African counties coloured by their Life Expectancy values*)

Issues:

- **Matching data points with regions on a map:** Mapping software typically draws regional maps using so-called **shapefiles** which contain the boundary of each region (as the vertices of a polygon) and the names of the regions. To produce a choropleth map, the shape file is used to draw the outlines of each region. Then a variable in the data set containing the names of the regions has to be matched with a corresponding naming-variable in the shape file so the program knows what observations in the data file correspond to what regions on the map.
    - Several sets of name variables are often included in a shape file (e.g. the set off full names and a parallel set of common abbreviations) to make it more likely that there will be a set of names that matches with the region-name variable in a data set. Otherwise the names in the data set will have to be changed to conform with one of the naming conventions available in the shape file.
- **Labels:** Viewers will seldom reliably know which regions are which just by looking at the map so name-labels are often added (e.g. the names of each state or their abbreviated form). The numeric value of the plotted variable might also be added for each region.

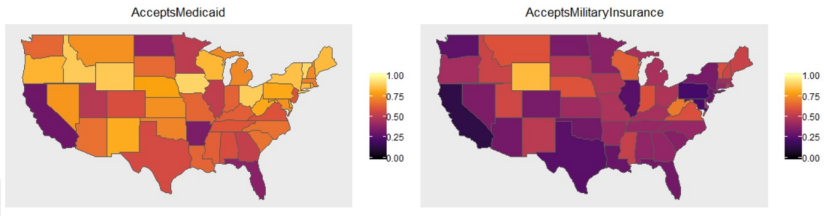


- **Scales:** Each map only displays the values of a single variable (in addition to region), but we can look at the behaviour of several variables at the same time if we put several maps on the same page, one for each variable.



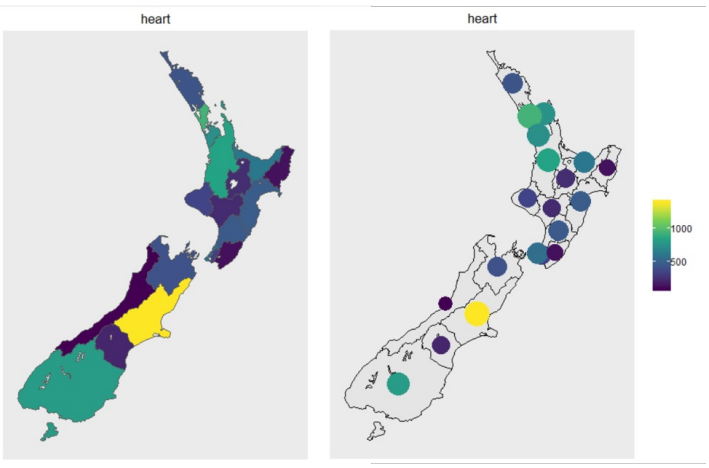
(*GDP per capita and Life Expectancy are measured on very different scales*)

- Then the issue of what to do about scales becomes important. Are the scales the variables are measured on comparable?
    - ***Non-comparable scales case*:** When the scales are not comparable (as with GDP per capita and Life Expectancy in the plots above) we translate values to colours in graphs independently of one another



(*One map shows the proportion of medical centres accepting Medicaid and the other accepting Military Insurance*)

- ***Comparable scales*:** But when the different variables are measured on the same scale (as in the plots above where both variables are proportions), we generally want to translate variable-values to colour values in the same way for all of the graphs so that we can compare values across graphs
- **Distortions due to land area:** Colouring regions misleadingly makes large unpopulated areas look more important than smaller highly populated areas (e.g. US political maps produced in this way misleadingly make the much-less-populated centre regions of the country look much more important contributors to election results than the heavily populated coastal regions)
    - **Alternatives** that try to get around this problem **:**
        - **Colour superimposed symbols rather that the regions themselves:** Use colour shapes (e.g. disks) centred at the centre of each region and size the shapes proportional to population (as in the "heart" example below). We call these disks ***centroids***.
        - **Cartograms:** distort the shapes in the map so the region sizes are approximately proportional to population sizes (*not implemented in iNZight yet*)
        - **Dots:** pepper regions with coloured dots, with each dot representing the same number of people (*not implemented in iNZight yet*)

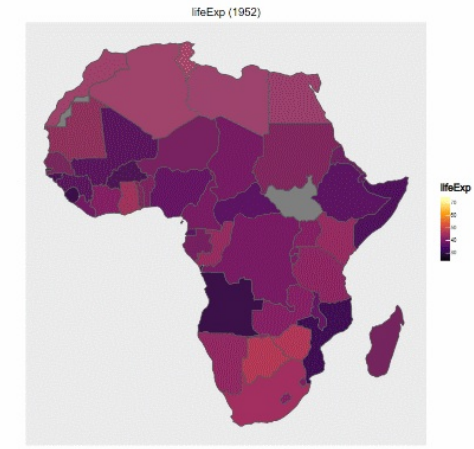


(*Shading versus population-sized centroids*)

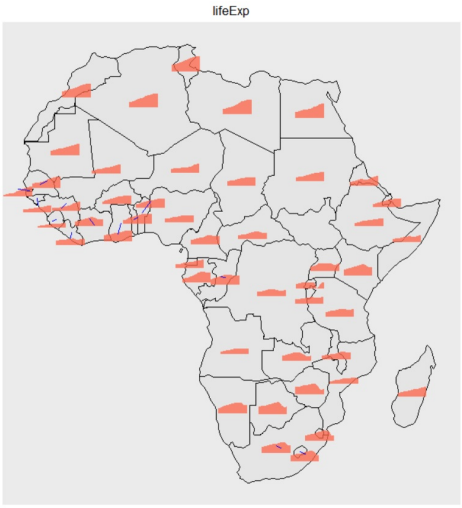
- **Conveying several variables on a single map**

- There are a variety of multivariate glyphs that can be used as superimposed objects (instead of colouring regions)

- **Changes over time (or values of some other variable):**
  - **Playing through a sequence of maps:** Can be good for looking at changes over time
  - **Superimpose glyphs** that are a mini time-series
    - Interactive graphs link to more detailed time series in various ways



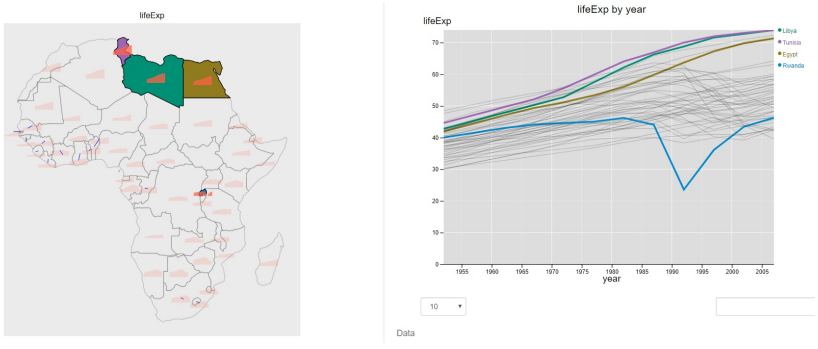
(***Playing through a sequence*** *of maps,* ***one for each year*** *- animated gif*)



(*Drawing a mini time-series for each country*)

Interactive Plots

Preview of Example (*still image*)



(*Linking countries and detailed time series (screenshot)*)

Example of an interactive plot for showing changes over time with time series

There is a great deal of interactivity besides hover-over in this plot. Clicking on a Country (but not on the little icon within the country) highlights the detailed time-series plot for that country on the right. Holding down the shift key while you click enables you to add countries. You can likewise hover over the set of time series on the right-hand plot to discover details, or click (or shift-click to accumulate countries) to link back to the map on the left. You can also pan (click and drag) and zoom (mouse wheel) on the map plot to get closer to small areas. (*Click the graphic's help page -- small "?" at top right.*)

Example of an interactive plot for showing changes over time using a "movie"

Click "Table" to remove the table. You can change the year using the slider at the upper-right, or play across years by clicking play button just under it. The variable being represented on the map can be changed using "Variable to display". Panning and zooming works here too. (*Click the graphic's help page -- small "?" at top right.*)

Some more issues

(*yet to discuss*)

- Map projections
- Colour palettes – what's good for what
- Compensating for deficiencies in colour perception – augmenting maps with complementary plots/tables

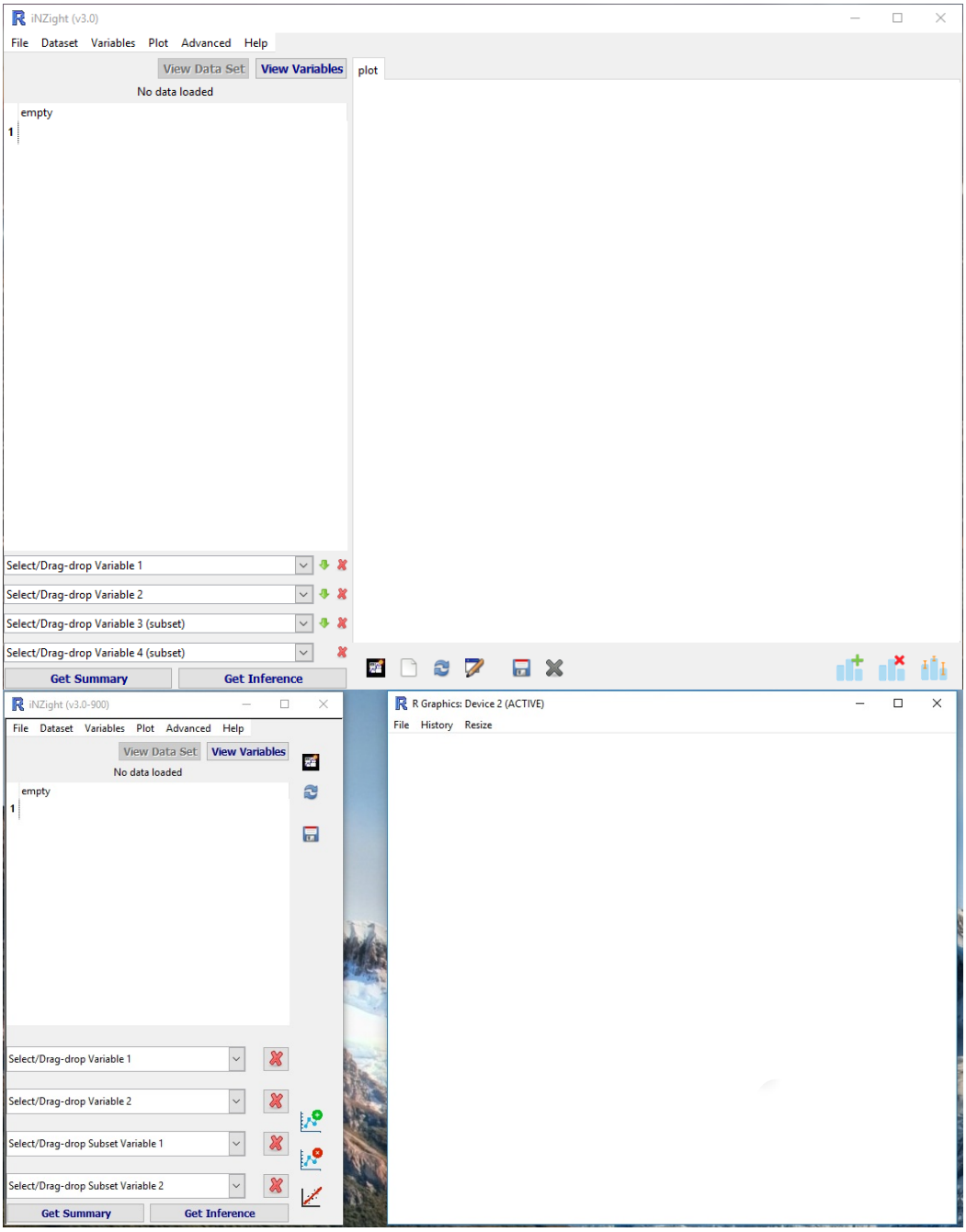
Extra Features

Once you're familiar with the basics of iNZight, you can try out some of the more advanced features. Some of them have been documented on this page.

- **iNZight Display Mode**
- **Colour Specification**
- **Export Interactive HTML Graphs**

iNZight Display Mode

iNZight (> 3.0) has two display modes: the traditional all-in-one window, as well as a separate graphics window and a narrow control panel (like VIT). The two are shown here:



To switch between the two, go to **File** > **Preferences**, and check the box **"Use dual-window display mode"**.

Colour Specification

In many of the Colour specification windows, you can either select from the menu, or type in a value of your own. Colours can specified by

the name of **any colour known by R**, or alternatively by using hexadecimal (HEX) codes.

To specify a HEX code, you enter a hash "#" followed by 6 digits (numbers 0-9 and/or letters A-F). The 6-digit HEX code is divided into three 2-digit codes that specify red, green, and blue, respectively. 00 corresponds to 0%, and FF corresponds to 100%. Some examples are:

- #000000 = black
- #FFFFFF = white
- #0000CC #009900 #FF0000 #c09040

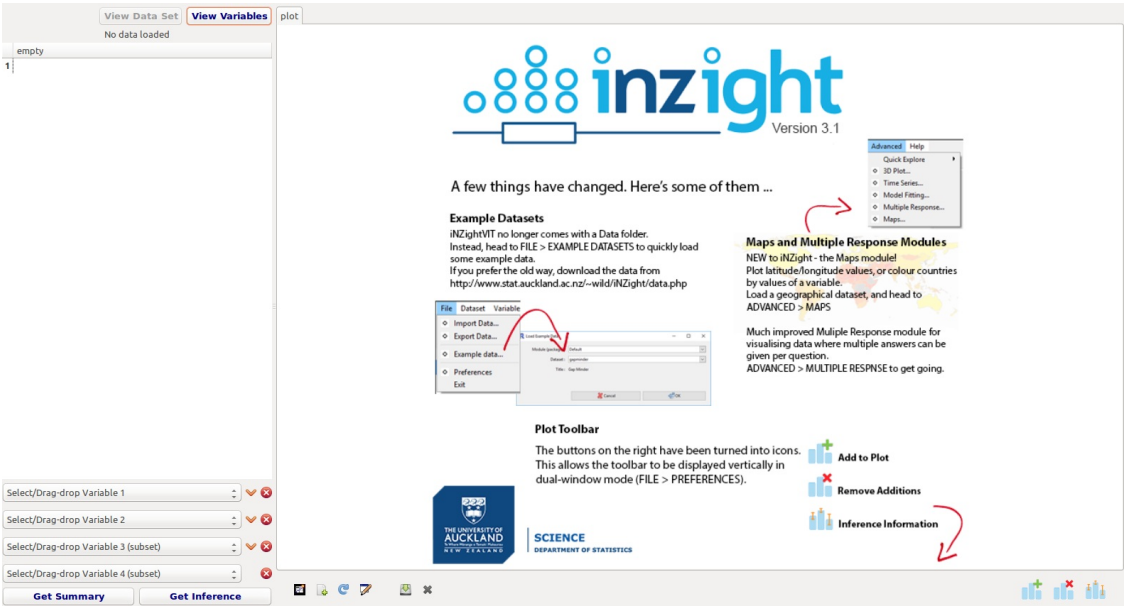
Opacity

You can also specify opacity when using HEX codes. This is done simply by adding two additional digits (forming an 8-digit code), where 00 = 0% opacity = completely invisible.

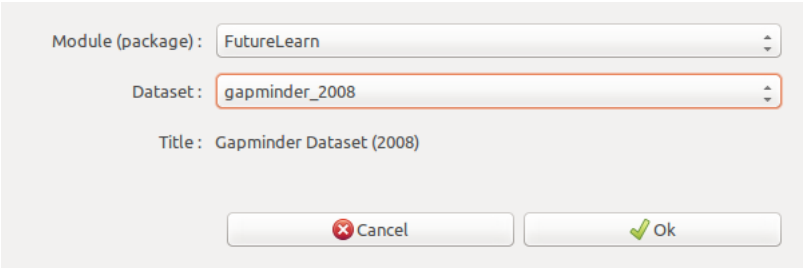
Export Interactive HTML Graphs developmental

A new feature of iNZight is the ability to export plots as SVG and Interactive HTML pages. It's still a work-in-progress that was part of a student's summer project. Please play around with it and let us know what you like, what doesn't work so well, and any other feedback you have!

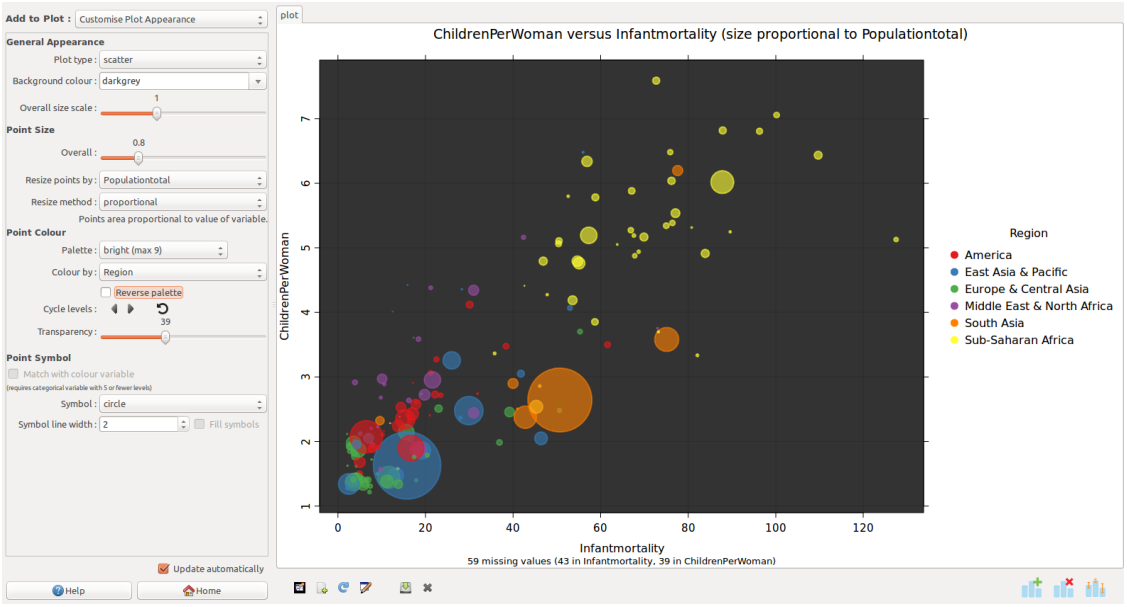
Let's load the `gapminder-2008` data from the FutureLearn module:



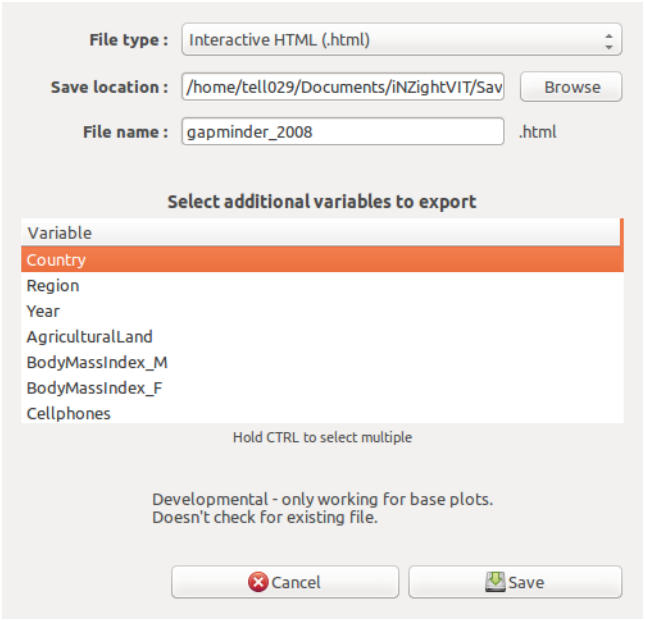
File > **Example Data** and select the Gapminder 2008 dataset:



Now plot some variables. Here we're using ChildrenPerWoman and Infantmortality, coloured by Region and sized by Populationtotal, but you can use whichever you prefer.



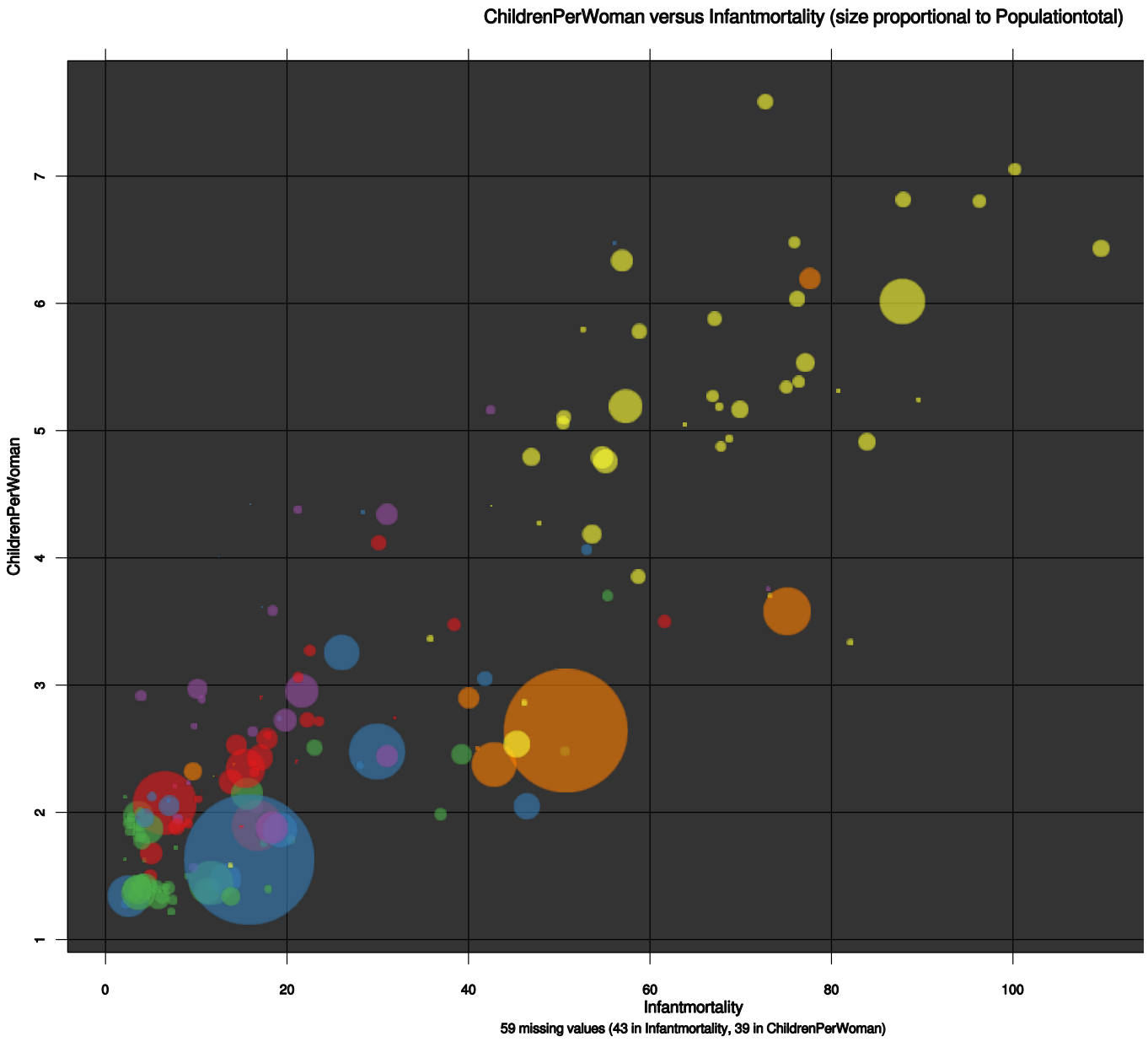
Now click the "Save" icon in the **plot toolbar**, or from the Plot menu.



- **File type**: choose "Interactive HTML"
- **Save location**: You can leave this as default, which is the iNZightVIT/Saved Plots folder in your Documents.
- **File name**: choose something appropriate
- **Additional variable**: We've selected "Country" as an additional variable to export, so we can use it to label points.

Once you click save, you might be prompted to install additional packages. Click "Yes" and wait while that happens, then you can save again.

The HTML file will open in your browser.



Variables to display

- Display all
- Country
- Infantmortality
- ChildrenPerWoman
- Region
- Populationtotal

Reset | View Table

- **Hover your mouse over points** to see details about that observation
- **Click points** to add to selection, and "View Table" to see the rows of the data set for those observations
- **Click-and-drag** to select a region of points
- **Click a legend label** to highlight points in that group

Click here to open the file