Package ‘iNZightPlots’

June 10, 2025

Type Package

Title Graphical Tools for Exploring Data with 'iNZight'

Version 2.16.0

Description Simple plotting function(s) for exploratory data analysis with flexible options allowing for easy plot customisation. The goal is to make it easy for beginners to start exploring a dataset through simple R function calls, as well as provide a similar interface to summary statistics and inference information. Includes functionality to generate interactive HTML-driven graphs. Used by 'iNZight', a graphical user interface providing easy exploration and visualisation of data for students of statistics, available in both desktop and online versions.

BugReports https://github.com/iNZightVIT/iNZightPlots/issues

Contact inzight_support@stat.auckland.ac.nz

URL https://inzight.nz

Depends R (>= 4.0)

Imports boot, chron, colorspace, dichromat, dplyr, emmeans, expss, grDevices, grid, hexbin, hms, iNZightMR (>= 2.2.7), iNZightTools (>= 1.9), lubridate, magrittr, quantreg, rlang, s20x, scales, stats, stringr, units, survey, utils

Suggests covr, forcats, DBI, dbplyr, ggbeeswarm, ggmosaic, ggplot2, ggridges, ggtext, ggthemes, gridSVG (>= 1.7-2), hextri, jsonlite, kableExtra, knitr, plotly, RColorBrewer, RSQLite, testthat, tibble, tidyr, viridis, waffle

License GPL-3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.1

Config/Needs/dependencies tmelliott/surveyspec@develop, iNZightVIT/iNZightTools@1.13.0, iNZightVIT/iNZightMR@2.2.7, rcmdcheck, curl

Config/Needs/coverage tmelliott/surveyspec@develop, iNZightVIT/iNZightTools@1.13.0, iNZightVIT/iNZightMR@2.2.7, covr

NeedsCompilation no

Author Tom Elliott [aut, cre] (ORCID: <https://orcid.org/0000-0002-7815-6318>),
Yu Han Soh [aut],
Daniel Barnett [aut],
Simon Anastasiadis [ctb] (Social Wellbeing Agency, NZ; RR3 method)

Maintainer Tom Elliott <tom.elliott@auckland.ac.nz>

Repository CRAN

Date/Publication 2025-06-09 22:40:02 UTC

Contents

`can.interact` *Identify if a plot can be interactive*

Description

Several iNZightPlots graphs have been enabled with custom interaction, while others make use of the automatic output of ‘plotly‘. This function returns ‘TRUE‘ if the provided plot has interaction (as determined by iNZight), and ‘FALSE‘ otherwise.

Usage

```
can.interact(x)

## Default S3 method:
can.interact(x)

## S3 method for class 'inzplotoutput'
can.interact(x)

## S3 method for class 'ggplot'
can.interact(x)
```

Arguments

`x`	a plot object returned from a plotting function

Details

Not that, while most ‘ggplot2‘ graphs can be passed to ‘plotly‘, and even though we are using plot.ly directly for some of our ggplot2 graphs, we still only return ‘TRUE‘ if the graph was created by one of the packages in the iNZight collection.

Value

Logical to identify if there is an interactive version

Methods (by class)

- `can.interact(default)`: Default interaction helper (always returns ‘FALSE‘)
- `can.interact(inzplotoutput)`: Graphs from ‘iNZightPlot()‘, many of which have interaction enabled, but some do not (for example, hex plots)
- `can.interact(ggplot)`: Those ‘iNZight*‘ plotting functions which return a ‘ggplot2‘ object and have been tested to work with plotly will be tagged as such; this is just a helper to check for the necessary attribute.

Author(s)

Tom Elliott, Yu Han Soh

Examples

```
can.interact(iNZightPlot(Sepal.Length, data = iris))
```

`construct_call` *Construct plot call from settings list*

Description

Construct plot call from settings list

Usage

```
construct_call(
  settings,
  vartypes,
  data = quote(.dataset),
  design = quote(.design),
  what = c("plot", "summary", "inference")
)
```

Arguments

`settings`	a list of plot settings, similar to `inzpar()`
`vartypes`	a list of variables types (numeric, factor)
`data`	a data set to pass to the call
`design`	a survey design (can be NULL)
`what`	the type of call to produce

Value

a plot/summary/inference call

`const_palette_names` *An incorrectly spelled function - deprecated*

Description

This function was misspelled in earlier versions and has been corrected to `cont_palette_names`, which should be used instead.

Usage

```
const_palette_names()
```

Value

a list of continuous colour palettes

See Also

`cont_palette_names`

`convert.to.factor` *Convert to Factor*

Description

Convert a numeric variable in to a factor with four levels.

Usage

```
convert.to.factor(x)
```

Arguments

`x`	a numeric vector

Value

a factor variable

Author(s)

Tom Elliott

Examples

```
f <- convert.to.factor(runif(100, 0, 10))
levels(f)
```

`create` *Create plots for iNZight*

Description

Create a Plot Object

Usage

```
create(obj, ...)
```

Arguments

`obj`	an object
`...`	additional arguments

Details

This create method is to be used by packages extending 'iNZightPlots', and should not be used by users. The resulting object should have an associated `plot` method.

Value

an iNZight plot object with class determined by data type

Author(s)

Tom Elliott

`emphasize_pal_colour` *Emphasize a level or interval of a colour palette*

Description

Emphasize a level or interval of a colour palette

Usage

```
emphasize_pal_colour(n, k, cat = TRUE, ncat = 5, fn)
```

Arguments

`n`	the number of colours to draw from the palette
`k`	the index of the colour to emphasize
`cat`	logical indicator if palette is categorical or numeric
`ncat`	the number of intervals to use for continuous palettes
`fn`	the colour palette function to use

Value

a colour palette, with one level emphasized (or range for numeric)

Author(s)

Tom Elliott

Examples

```
pal <- inzpalette("bright")
plot(1:5, pch = 19, col = emphasize_pal_colour(5, 2, fn = pal))
```

`exploreAllPlots` *Explore all Univariate Plots*

Description

Allows easy viewing of every variable in the data set. The user will be prompted to see the next variable.

Usage

```
exploreAllPlots(data)
```

Arguments

`data`	a data frame

Author(s)

Tom Elliott

Examples

```
if (interactive())
    exploreAllPlots(iris)
```

`exploreAllSummaries` *Explore all Univariate Summaries*

Description

Allows easy access to a summary for every variable in the data set.

Usage

```
exploreAllSummaries(data, ...)

## S3 method for class 'allSummaries'
print(x, ...)
```

Arguments

`data`	a data set
`...`	additional arguments passed to getPlotSummary()
`x`	an `allSummaries` object

Value

allSummaries object, a concatenation of summaries from all variables

Functions

- `print(allSummaries)`: print method for allSummaries object

Author(s)

Tom Elliott

Examples

```
exploreAllSummaries(iris)
```

`exportHTML` *ExportHTML*

Description

`exportHTML` is designed to export the iNZight plot as a dynamic, interactive HTML page. Currently only handles single panel plots. Coloured hex plots are currently not available yet.

Usage

```
exportHTML(
  x,
  file = file.path(dir, "index.html"),
  data,
  local = FALSE,
  dir = tempdir(),
  extra.vars,
  ...
)

## S3 method for class '`function`'
exportHTML(
  x,
  file = file.path(dir, "index.html"),
  data = NULL,
  local = FALSE,
  dir = tempdir(),
  extra.vars = NULL,
  width = dev.size()[1],
  height = dev.size()[2],
  ...
)
```

```
## S3 method for class 'ggplot'
exportHTML(
  x,
  file = file.path(dir, "index.html"),
  data = NULL,
  local = FALSE,
  dir = tempdir(),
  extra.vars = NULL,
  mapObj,
  ...
)

## S3 method for class 'inzplotoutput'
exportHTML(
  x,
  file = file.path(dir, "index.html"),
  data = NULL,
  local = FALSE,
  dir = tempdir(),
  extra.vars = NULL,
  ...
)
```

Arguments

`x`	An iNZight plot object that captures iNZight environment
`file`	Name of temporary HTML file generated (defaults to 'index.html' in a temporary directory, or other as specified using 'dir')
`data`	dataset/dataframe that you wish to investigate and export more variables from
`local`	Logical for creating local files for offline use (default to false)
`dir`	A directory to store the file and output
`extra.vars`	extra variables specified by the user to be exported
`...`	extra arguments
`width`	the desired width of the SVG plot
`height`	the desired height of the SVG plot
`mapObj`	iNZightMap object (from iNZightMaps)

Value

an inzHTML object consisting of a link to an HTML rendering of `x` with filename `file`, which can be loaded in the browser (for example using `browseURL`, or calling the `print()` method of the returned object.

Methods (by class)

- `exportHTML(`function`)`: method for an iNZightPlot-generating function
- `exportHTML(ggplot)`: method for iNZightMaps or other supported ggplot graphs
- `exportHTML(inzplotoutput)`: method for output from iNZightPlot

Author(s)

Yu Han Soh

Examples

```
## Not run:
x <- iNZightPlot(Petal.Width, Petal.Length, data = iris, colby = Species)
exportHTML(x, "index.html")

#to export more variables for scatterplots:
 exportHTML(x, "index.html", data = iris, extra.vars = c("Sepal.Length", "Sepal.Width"))

## End(Not run)
```

---

exportSVG *Export iNZightPlots as an SVG*

Description

`exportSVG` is designed to export the iNZight plot as a temporary SVG that is opened in a web browser. The iNZightPlot must be drawn to a graphics device before exporting can occur.

Usage

```
exportSVG(x, file = tempfile(fileext = ".svg"), ...)

## S3 method for class '`function`'
exportSVG(
  x,
  file = tempfile(fileext = ".svg"),
  width = dev.size()[1],
  height = dev.size()[2],
  ...
)

## S3 method for class 'inzplotoutput'
exportSVG(x, file = tempfile(fileext = ".svg"), ...)
```

Arguments

`x`	iNZight plot object or function that captures iNZight environment
`file`	Name of temporary svg file generated (by default: 'inzightplot.svg')
`...`	additional arguments
`width`	the width of the plot device
`height`	the height of the plot device

Value

Opens up an SVG file of `x` with filename `file` in a web browser

Methods (by class)

- `` exportSVG(`function`) ``: method for functions
- `exportSVG(inzplotoutput)`: method for an existing plot object

Author(s)

Yu Han Soh

`getPlotSummary` *iNZight Plot Summary and Inference*

Description

Generate summary or inference information for an iNZight plot

Usage

```
getPlotSummary(
  x,
  y = NULL,
  g1 = NULL,
  g1.level = NULL,
  g2 = NULL,
  g2.level = NULL,
  varnames = list(),
  colby = NULL,
  sizeby = NULL,
  data = NULL,
  design = NULL,
  freq = NULL,
  missing.info = TRUE,
  inzpars = inzpar(),
  summary.type = "summary",
  table.direction = c("horizontal", "vertical"),
  hypothesis.value = 0,
  hypothesis.alt = c("two.sided", "less", "greater"),
  hypothesis.var.equal = FALSE,
  hypothesis.use.exact = FALSE,
  hypothesis.test = c("default", "t.test", "anova", "chi2", "proportion"),
  hypothesis.simulated.p.value = FALSE,
 hypothesis = list(value = hypothesis.value, alternative = match.arg(hypothesis.alt),
    var.equal = hypothesis.var.equal, use.exact = hypothesis.use.exact, test =
   match.arg(hypothesis.test), simulated.p.value = hypothesis.simulated.p.value),
  survey.options = list(),
  width = 100,
  epi.out = FALSE,
  privacy_controls = NULL,
  html = FALSE,
  ...,
  env = parent.frame()
)
```

Arguments

`x`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object
`y`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object
`g1`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object. This variable acts as a subsetting variable.
`g1.level`	the name (or numeric position) of the level of `g1` that will be used instead of the entire data set
`g2`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object. This variable acts as a subsetting variable, similar to `g1`
`g2.level`	same as `g1.level`, however takes the additional value `"_MULTI"`, which produces a matrix of `g1` by `g2`
`varnames`	a list of variable names, with the list named using the appropriate arguments (i.e., `list(x = "height", g1 = "gender")`)
`colby`	the name of a variable (numeric or factor) to colour points by. In the case of a numeric variable, a continuous colour scale is used, otherwise each level of the factor is assigned a colour
`sizeby`	the name of a (numeric) variable, which controls the size of points
`data`	the name of a data set
`design`	the name of a survey object, obtained from the `survey` package
`freq`	the name of a frequency variable if the data are frequencies
`missing.info`	logical, if `TRUE`, information regarding missingness is displayed in the plot
`inzpars`	allows specification of iNZight plotting parameters over multiple plots
`summary.type`	one of `"summary"` or `"inference"`

`table.direction`	one of 'horizontal' (default) or 'vertical' (useful for many categories)
`hypothesis.value`	H0 value for hypothesis test
`hypothesis.alt`	alternative hypothesis (!=, <, >)
`hypothesis.var.equal`	use equal variance assumption for t-test?
`hypothesis.use.exact`	logical, if `TRUE` the exact p-value will be calculated (if applicable)
`hypothesis.test`	in some cases (currently just two-samples) can perform multiple tests (t-test or ANOVA)
`hypothesis.simulated.p.value`	also calculate (where available) the simulated p-value
`hypothesis`	either NULL for no test, or missing (in which case above arguments are used)
`survey.options`	additional options passed to survey methods
`width`	width for the output, default is 100 characters
`epi.out`	logical, if `TRUE`, then odds/rate ratios and rate differences are printed when appropriate (`y` with 2 levels)
`privacy_controls`	optional, pass in confidentialisation and privacy controls (e.g., random rounding, suppression) for microdata
`html`	logical, it `TRUE` output will be returned as an HTML page (if supported)
`...`	additional arguments, see `inzpar`
`env`	compatibility argument

Details

Works much the same as `iNZightPlot`

Value

an `inzight.plotsummary` object with a print method

Author(s)

Tom Elliott

Examples

```
getPlotSummary(Species, data = iris)
getPlotSummary(Species, data = iris,
    summary.type = "inference", inference.type = "conf")

# perform hypothesis testing
getPlotSummary(Sepal.Length, data = iris,
    summary.type = "inference", inference.type = "conf",
    hypothesis.value = 5)

# if you prefer a formula interface
inzsummary(Sepal.Length ~ Species, data = iris)
inzinference(Sepal.Length ~ Species, data = iris)

## confidentialisation and privacy controls
# random rounding and suppression:
HairEyeColor_df <- as.data.frame(HairEyeColor)
inzsummary(Hair ~ Eye, data = HairEyeColor_df, freq = Freq)
inzsummary(Hair ~ Eye, data = HairEyeColor_df, freq = Freq,
    privacy_controls = list(
        rounding = "RR3",
        suppression = 10
    )
)
```

`iNZightPlot` *iNZight Plot*

Description

A general plotting function that automatically detects variable type and draws the appropriate plot. It also provides facilities to add inference information to plots, colour- and size-by variables, and can handle survey data.

Usage

```
iNZightPlot(
  x,
  y = NULL,
  g1 = NULL,
  g1.level = NULL,
  g2 = NULL,
  g2.level = NULL,
  varnames = list(),
  colby = NULL,
  sizeby = NULL,
  symbolby = NULL,
  extra.vars,
  locate = NULL,
  locate.id = NULL,
  locate.col = NULL,
  locate.extreme = NULL,
  locate.same.level = NULL,
  highlight = NULL,
  data = NULL,
  design = NULL,
  freq = NULL,
  missing.info = TRUE,
  xlab,
  ylab,
  show_units = TRUE,
  new = TRUE,
  inzpars = inzpar(),
  layout.only = FALSE,
  plot = TRUE,
  xaxis = TRUE,
  yaxis = TRUE,
  xlim = NULL,
  ylim = NULL,
  zoombars = NULL,
  hide.legend = FALSE,
  df,
  env = parent.frame(),
  ...
)
```

Arguments

`x`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object
`y`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object
`g1`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object. This variable acts as a subsetting variable.
`g1.level`	the name (or numeric position) of the level of `g1` that will be used instead of the entire data set
`g2`	a vector (numeric or factor), or the name of a column in the supplied `data` or `design` object. This variable acts as a subsetting variable, similar to `g1`
`g2.level`	same as `g1.level`, however takes the additional value `"_MULTI"`, which produces a matrix of `g1` by `g2`
`varnames`	a list of variable names, with the list named using the appropriate arguments (i.e., `list(x = "height", g1 = "gender")`)
`colby`	the name of a variable (numeric or factor) to colour points by. In the case of a numeric variable, a continuous colour scale is used, otherwise each level of the factor is assigned a colour
`sizeby`	the name of a (numeric) variable, which controls the size of points
`symbolby`	the name of a factor variable to code point symbols
`extra.vars`	the names of any additional variables to be passed through the internal functions to the create and plot methods.
`locate`	variable to label points

`locate.id`	id of points (row numbers) to label, or an expression that evaluates as a logical vector (e.g., `x > 5`)
`locate.col`	the colour to locate points if a variable is not specified
`locate.extreme`	`numeric`, the number of extreme points to label (using Mahalanobis' distance)
`locate.same.level`	name of a variable to label points with same level of as those specified with 'locate.id'
`highlight`	`numeric` vector consisting of the row numbers/IDs of points to highlight
`data`	the name of a data set
`design`	the name of a survey object, obtained from the `survey` package
`freq`	the name of a frequency variable if the data are frequencies
`missing.info`	logical, if `TRUE`, information regarding missingness is displayed in the plot
`xlab`	the text for the x-label
`ylab`	the text for the y-label
`show_units`	logical, if 'TRUE' (default) units will be shown beside axies and legend variable labels
`new`	logical, used for compatibility
`inzpars`	allows specification of iNZight plotting parameters over multiple plots
`layout.only`	logical, if `TRUE`, only the layout is drawn (useful if a custom plot is to be drawn)
`plot`	logical, if `FALSE`, the plot is not drawn (used by `summary`)
`xaxis`	logical, whether or not to draw the x-axis
`yaxis`	logical, whether or not to draw the y-axis
`xlim`	specify the x limits of the plot
`ylim`	specify the y limits of the plot
`zoombars`	numeric, length 2; when drawing a bar plot, if the number of bars is too large, the user can specify a subset. The first value is the starting point (1 is the first bar, etc), while the second number is the number of bars to show.
`hide.legend`	logical, if TRUE, the `legend` will not be drawn
`df`	compatibility argument
`env`	compatibility argument
`...`	additional arguments, see `inzpar`

Details

The main goal of 'iNZightPlots' is to make it easy to beginners to explore a dataset graphically, using a suite of simple arguments to add features to their graph.

The second use of this function is within the companion software 'iNZight', providing a single function call with arguments controlled by the user through a GUI.

Value

An `inzightplotoutput` object, which contains the information displayed in the plot

Author(s)

Tom Elliott

Examples

```
iNZightPlot(Species, data = iris)
iNZightPlot(Petal.Width, g1 = Species, data = iris)

iNZightPlot(Sepal.Length, Sepal.Width, data = iris,
    colby = Species)
iNZightPlot(Sepal.Length, Sepal.Width, data = iris,
    colby = Species, trend = c("linear", "quadratic"),
    trend.by = TRUE, trend.parallel = FALSE)

# add inference information
iNZightPlot(Petal.Width, data = iris,
    inference.type = "conf", inference.par = "mean")
iNZightPlot(Petal.Width, data = iris,
    inference.type = "conf", inference.par = "mean",
    bootstrap = TRUE)

# alternatively, use the formula interface
inzplot(Sepal.Length ~ Sepal.Width | Species, data = iris)
```

`inzinference` *iNZight Inference Method*

Description

A generic function used to generate inferential information for objects within the iNZight ecosystem.

Usage

```
inzinference(x, ..., env = parent.frame())

## S3 method for class 'formula'
inzinference(
  x,
  data = NULL,
  design = NULL,
  type = c("conf", "comp"),
  ...,
  env = parent.frame()
)
```

Arguments

`x`	An object
`...`	additional arguments for methods
`env`	an environment to evaluate things
`data`	Dataset to plotq
`design`	A survey design to use
`type`	Type type of inference to obtain, one of 'conf' or 'comp' for confidence intervals and comparison intervals, respectively (currently ignored).

Value

The output depends on the type of input, and consists of a inference object with a `print` method.

Methods (by class)

- `inzinference(formula)`: Wrapper for getPlotSummary to obtain inference information about a plot

---

`inzpalette` *iNZight colour palette*

Description

Used to obtain a colour palette of a given name. A list of available palettes can be obtained by ‘cat_palette_names()‘ and ‘cont_palette_names()‘.

Usage

```
inzpalette(palette)

cat_palette_names()

cont_palette_names()
```

Arguments

`palette`	the name of a palette

Value

a colour palette function with single argument ‘n‘

Functions

- `cat_palette_names()`: List of categorical colour palettes
- `cont_palette_names()`: List of continuous colour palettes

Author(s)

Tom Elliott

Examples

```
plot(1:5, pch = 19, col = inzpalette("bright")(5))

# for a list of palette names
cat_palette_names()
cont_palette_names()
```

`inzpar` *iNZight Plotting Parameters*

Description

Plotting parameters for iNZight Plots

Usage

```
inzpar(..., .viridis = requireNamespace("viridis", quietly = TRUE))
```

Arguments

`...`	If arguments are supplied, then these values are set. If left empty, then
`.viridis`	checks if the viridis package is installed; or can be turned off the default list is returned.

Details

A whole suite of parameters that can be used to fine-tune plots obtained from the `iNZightPlot` function. The parameters include both plot type, style, and appearance.

'pch' the plotting symbol to be used; default is '21' (circle with fill)

'col.pt' the colour of points. this can either be a single value, or a vector of colours if `colby` is specified

'col.fun' a function to use for colouring points, etc., or the name of a palette, see `inzpalette`

'col.emph', 'col.emphn' emphasize the chosen level of a colour by variable. For numeric colour by, `col.emphn` specifies the number of quantiles to use.

'emph.on.top' if `TRUE`, emphasised points will be positioned on top

'col.default' the default colour functions, containing a list with entries for 'cat' and 'cont' variables

'col.missing' the colour for missing values; default is a light grey

'reverse.palette' logical, if `TRUE` the palette will be reversed

'col.method' the method to use for colouring by a variable, one of 'linear' or 'rank'

'cex' the overall scaling for the entire plot; values less than 1 will make the text and points smaller, while values larger than 1 will magnify everything

'cex.pt' the scaling value for points

'cex.dotpt' the scaling value for points in a dotplot. Note, this is not multiplicative with `'cex.pt'`

'cex.lab' the scaling value for the plot labels

'cex.axis' the scaling value for the axis labels

'cex.main' the scaling value for the main plot title

'cex.text' the scaling value for text on the plot

'resize.method' one of 'proportional' (default) or 'emphasize'

'alpha' transparency setting for points; default is 1, 0 is fully transparent

'bg' the background colour for the plot

'grid.lines' logical to control drawing of axis grid lines

'col.grid' if 'grid.lines' is `TRUE`, this controls the colour of them. The default is 'default', which will choose a colour based on the value of 'bg')

'fill.pt' the fill colour for points; default is `"transparent"`

'lwd' the line width of lines (for joining points)

'lty' the line type of lines (for joining points)

'lwd.pt' the line width used for points; default is 2

'col.line' the colour of lines used to join points

'col.sub' vector of up to two colours for the background of subplot labels. If only one specified, it is used for both.

'locate.col.def' the default colour for locating points

'highlight.col' colour to use for highlighting points

'jitter' the axes to add jitter to. Takes values `"x"`, `"y"`, or `"xy"` (default is en empty string, `""`)

'rugs' the axes to add rugs to. Takes same values as `jitter`

'trend' a vector containing the trend lines to add to the plot. Possible values are `c("linear", "quadratic", "cubic")`

'smooth' the smoothing (lowess) for the points. Takes a value between 0 and 1 (the default, 0, draws no smoother)

'smoothby.lty' the line type used for smoothers if `trend.by = TRUE`

'quant.smooth' if quantile smoothers are desired, they can be specified here as either the quantiles to smooth over (e.g., `c(0.25, 0.5, 0.75)`), or `"default"`, which uses the sample size to decide on an appropriate set of quantile smoothers

'LOE' logical, if `TRUE`, then a 1-1 line of equality is drawn

'join' logical, if `TRUE`, then points are joined by lines

'lines.by logical, if `join = TRUE` and `colby` is specified, points are joined by the specified variable

'col.trend' a named list of colours to be used for drawing the lines. The default is `list(linear = "blue", quadratic = "red", cubic = "green4")`

'lty.trend' a named list of line types for various types of trend lines. The default is `list(linear = 1, quadratic = 2, cubic = 3)`

'trend.by' logical, if `TRUE`, then trend lines are drawn separately for each group specified by `colby`

'trend.parallel' logical, if `TRUE`, the trend lines by group are given the same slope; otherwise they are fit independently

'col.smooth' the colour of the smoother

'col.LOE' the colour of the line of equality

'lty.LOE' the line type of the line of equality

'boxplot' logical, if `TRUE`, a boxplot is drawn with dotplots and histograms

'box.lwd', 'box.col', 'box.fill' the line width, colour, and fill colour for the box plot drawn

'bar.lwd', 'bar.col', 'bar.fill' the line width, colour, and fill colour of bars in a bar plot

'bar.counts' logical, if `TRUE` bar graphs will display counts instead of percentages (the default)

'bar.relative.width' logical, if `TRUE` the width of bars will be proportional to the number of observations in each group (colour)

'full.height' may no longer be necessary ...

'inf.lwd.comp', 'inf.lwd.conf' the line width of comparison and confidence intervals, respectively

'inf.col.comp', 'inf.col.conf' the colour of comparison and confidence intervals, respectively. These take a length 2 vector, where the first element is used for normal inference, while the second is used for bootstrap intervals

'inference.type' the type of inference added to the plot. Possible values are `c("comp", "conf")`

'inference.par' the parameter which we obtain intervals for. For a dotplot or histogram, this can be either `"mean"` or `"median"`; for bar plots it can be "proportion"

'ci.width' the width of confidence intervals, default 0.95 for a 95% confidence interval

'bs.inference' logical, if `TRUE`, then nonparametric bootstrap simulation is used to obtain the intervals

'min.count' the min count for barplots inference; counts less than this are ignored

'n.boot' the number of bootstrap simulations to perform

'large.sample.size' sample sizes over this value will use a large-sample plot variant (i.e., scatter plots will become hex plots, dot plots become histograms)

'largesample' logical, if `TRUE`, then the large-sample plot variance is used

'scatter.grid.bins' the number, N, of bins to use for the scatter-grid plot, producing an N x N matrix

'hex.bins' the number of bins to use for hexagonal binning

'hex.style' the style of the hexagons, one of "size" or "alpha"

'hex.diffuse' logical, Pass on rounding error to nearest not-yet-drawn hexes so that rare classes get represented

'hist.bins' the number of bins to use for the histogram (The default `NULL` uses point size to approximate dot plot)

'quant.cutoff' if `quant.smooth = "default"`, these sample size values are used to determine which quantiles are drawn

'plottype' used to override the default plot type. Possible values, depending on data type, include `c("scatter"|"grid"|"hex"|"dot"|"hist")`

'matchplots' logical, if `TRUE`, then the type of plot is kept consistent between different subsets

'match.limits' a vector of two values used to decide whether to use all small-sample or all large-sample plots

'xlim' a vector defining the x axis limits (default NULL will use the data)

'ylim' a vector defining the y axis limits (default NULL will use the data)

'transform' a list of variable transformations (e.g., list(x = 'log'))

'plot.features' a list containing any additional features for new plots (e.g., maptype)

'round' integer specifying optional rounding of numerical output, default NA (ignored)

'round_percent' integer specifying rounding for percentages (default 2)

'signif' integer specifying number of significant figured in numeric output (default 2). Ignored if `round` is not NA.

Value

an object of class `inzpar.list`

Examples

```
# arguments can be passed directly to \code{iNZightPlot}
iNZightPlot(Sepal.Length,
    data = iris, col.pt = "red",
    box.col = "blue", box.fill = "green"
)

# or stored and passed to it (only pars relevant to the current
# plot are used)
mypar <- inzpar(
    col.pt = "red", box.col = "blue", box.fill = "green",
    trend = "linear", trend.by = TRUE
)
inzplot(Sepal.Length ~ Species, data = iris, inzpar = mypar)
iNZightPlot(Sepal.Length, Sepal.Width,
    data = iris, inzpar = mypar,
    colby = Species
)
```

`inzplot` *iNZight Plot Method*

Description

A generic function used to plot objects within the iNZight ecosystem.

Usage

```
inzplot(x, ..., env = parent.frame())

## S3 method for class 'formula'
inzplot(x, data = NULL, design = NULL, ..., env = parent.frame())
```

Arguments

`x`	A formula in the form of `y ~ x	g`. See Details.
`...`	Any arguments to pass to `iNZightPlot`
`env`	the parent environment to pass to the plot function
`data`	Dataset to plotq
`design`	A survey design to use

Details

`inzplot` is a simple wrapper around the `iNZightPlot` function.

There are four options for the formula passed in:

`y` will produce a plot of the single variable `y`.

`y ~ x` will produce a plot of `y` against `x`.

`y ~ x | g1` will produce a plot of `y` against `x` subset by `g1`.

`y ~ x | g1 + g2` will produce a plot of `y` against `x` subset by `g1` and `g2`.

Value

The output depends on the type of input, but is usually called for the side-effect of producing a plot.

An `inzightplotoutput` object, which contains the information displayed in the plot

See Also

iNZightPlot

Examples

```
data("CO2")
inzplot(~uptake, data = CO2)
inzplot(uptake ~ Treatment, data = CO2)
inzplot(uptake ~ Treatment | Type, data = CO2)
inzplot(uptake ~ Treatment | Type,
    data = CO2, g1.level = "Quebec"
)
```

`inzsummary` *iNZight Summary Method*

Description

A generic function used to summarize objects within the iNZight ecosystem.

Usage

```
inzsummary(x, ..., env = parent.frame())

## S3 method for class 'formula'
inzsummary(x, data = NULL, design = NULL, ..., env = parent.frame())
```

Arguments

`x`	An object
`...`	additional arguments for methods
`env`	an environment to evaluate things
`data`	Dataset to plotq
`design`	A survey design to use

Value

The output depends on the type of input, and consists of a summary object with a `print` method.

Methods (by class)

- `inzsummary(formula)`: Wrapper for getPlotSummary to obtain summary information about a plot

---

`mend_call` *Mend a plot call based on valid parameters*

Description

Mend a plot call based on valid parameters

Usage

```
mend_call(call, data, design_name, plot)
```

Arguments

`call`	a plot call string, or expression
`data`	the dataset
`design_name`	name of the design, if any
`plot`	the result of `inzplot`, `inzsummary`, or `inzinference`

Value

a plot call with extraneous arguments removed

print.inzHTML *Print method for ‘inzHTML‘ object*

Description

The default action is for the URL to be ’printed’ (opened) in the browser, unless ‘viewer‘ is specified as something else. If ‘viewer = NULL‘, then the URL is printed as a character string.

Usage

```
## S3 method for class 'inzHTML'
print(x, viewer = getOption("viewer", utils::browseURL), ...)
```

Arguments

`x`	a URL that will be printed
`viewer`	the viewing function to use to display the URL
`...`	additional arguments

Value

NULL (it’s a print function, after all)

snz_privacy_controls *Statistics New Zealand Privacy Controls*

Description

Based off Microdata Output Guide 2020 v5-1

Usage

```
snz_privacy_controls(type = c("survey"), weighted = type == "survey", ...)
```

Arguments

`type`	the type of data, used to specify the correct rules. Currently only survey (4.0.1) data is supported.
`weighted`	logical indicating if the results are a weighted survey design or not.
`...`	additional arguments, used to override defaults

Value

a list of privacy control rules

Index