Package 'pivottabler'

September 20, 2019

Type Package

Title Create Pivot Tables in R

Version 1.2.2

Description Create regular pivot tables with just a few lines of R. More complex pivot tables can also be created, e.g. pivot tables with irregular layouts, multiple calculations and/or derived calculations based on multiple data frames. Pivot tables are constructed using R only and can be written to a range of output formats (plain text, 'HTML', 'Latex' and 'Excel'), including with styling/formatting.

Depends R (>= 3.3.0)

Imports R6 (>= 2.2.0), dplyr (>= 0.5.0), data.table (>= 1.10.0), jsonlite (>= 1.1), htmltools(>= 0.3.5), htmlwidgets (>= 0.8)

Suggests ggplot2 (>= 2.2.0), lubridate (>= 1.5.0), listviewer (>= 1.4.0), openxlsx (>= 4.0.17), basictabler (>= 0.2.0), shiny, knitr, rmarkdown, testthat

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/cbailiss/pivottabler

BugReports https://github.com/cbailiss/pivottabler/issues

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Christopher Bailiss [aut, cre]

Maintainer Christopher Bailiss <cbailiss@gmail.com>

Repository CRAN

Date/Publication 2019-09-20 11:50:02 UTC

R topics documented:

`bhmtraindisruption` *Birmingham Train Disruptions, Dec 2016-Feb 2017.*

Description

A dataset containing details of the trains in the bhmtrains dataset that were partially/wholly cancelled and/or reinstated.

Usage

```
bhmtraindisruption
```

Format

A data frame with 2982 rows and 10 variables:

ServiceId Unique train identifier

LastCancellationAt Time of the last cancellation

LastCancellationLocation 3-letter code denoting the location of the last cancellation

LastCancellationReasonCategory The broad reason why the train was cancelled

LastCancellationReasonDesc A more specific reason why the train was cancelled

LastReinstatedAt Time of the last reinstatement

LastChangeOfOriginAt Time of the last change of origin

LastChangeOfOriginLocation 3-letter code denoting the location of the last change of origin

LastChangeOfOriginReasonCategory The broad reason why the origin was changed

LastChangeOfOriginReasonDesc A more specific reason why the origin was changed

Source

http://www.recenttraintimes.co.uk/

`bhmtrains` *Birmingham Trains, Dec 2016-Feb 2017.*

Description

A dataset containing all of the trains that either originated at, passed through or terminated at Birmingham New Street railway station in the UK between 1st December 2016 and 28th February 2017 inclusive.

Usage

```
bhmtrains
```

Format

A data frame with 83710 rows and 16 variables:

ServiceId Unique train identifier

Status Train status: A=Active, C=Cancelled, R=Reinstated

TOC Train operating company

TrainCategory Express Passenger or Ordinary Passenger

PowerType DMU=Diesel Multiple Unit, EMU=Electrical Multiple Unit, HST=High Speed Train

SchedSpeedMPH Scheduled maximum speed (in miles per hour)

Origin 3-letter code denoting the scheduled origin of the train

OriginGbttDeparture Scheduled departure time in the Great Britain Train Timetable (GBTT) from the origin station

OriginActualDeparture Actual departure time from the origin station

GbttArrival Scheduled arrival time in Birmingham in the GBTT

ActualArrival Actual arrival time in Birmingham in the GBTT

GbttDeparture Scheduled departure time from Birmingham in the GBTT

ActualDeparture Actual departure time from Birmingham in the GBTT

Destination 3-letter code denoting the scheduled destination of the train

DestinationGbttArrival Scheduled arrival time in the GBTT at the destination station

DestinationActualArrival Actual arrival time at the destination station

Source

http://www.recenttraintimes.co.uk/

checkArgument *Perform basic checks on a function argument.*

Description

checkArgument is a utility function that provides basic assurances about function argument values and generates standardised error messages when invalid values are encountered.

Usage

```
checkArgument(argumentCheckMode, checkDataTypes, className, methodName,
  argumentValue, isMissing, allowMissing = FALSE, allowNull = FALSE,
  allowedClasses = NULL, mustBeAtomic = FALSE,
  allowedListElementClasses = NULL, listElementsMustBeAtomic = FALSE,
  allowedValues = NULL, minValue = NULL, maxValue = NULL,
  maxLength = NULL)
```

Arguments

argumentCheckMode	A number between 0 and 4 specifying the checks to perform.
checkDataTypes	A logical value specifying whether the data types should be checked when argumentCheckMode=3
className	The name of the calling class, for inclusion in error messages.
methodName	The name of the calling method, for inclusion in error messages.
argumentValue	The value to check.
isMissing	Whether the argument is missing in the calling function.
allowMissing	Whether missing values are permitted.
allowNull	Whether null values are permitted.
allowedClasses	The names of the allowed classes for argumentValue.
mustBeAtomic	Whether the argument value must be atomic.
allowedListElementClasses	For argument values that are lists(), the names of the allowed classes for the elements in the list.
listElementsMustBeAtomic	For argument values that are lists(), whether the list elements must be atomic.
allowedValues	For argument values that must be one value from a set list, the list of allowed values.
minValue	For numerical values, the lowest allowed value.
maxValue	For numerical values, the highest allowed value.
maxLength	For character values, the maximum allowed length (in characters) of the value.

Value

No return value. If invalid values are encountered, the `stop()` function is used to interrupt execution.

`cleanCssValue` *Cleans up a CSS attribute value.*

Description

`cleanCssValue` is a utility function that performs some basic cleanup on CSS attribute values. Leading and trailing whitespace is removed. The CSS values "initial" and "inherit" are blocked. The function is vectorised so can be used with arrays.

Usage

```
cleanCssValue(cssValue)
```

Arguments

`cssValue`	The value to cleanup.

Value

The cleaned value.

`containsText` *Check whether a text value is present in another text value.*

Description

`containsText` is a utility function returns TRUE if one text value is present in another. Case sensitive. If textToSearch is a vector, returns TRUE if any element contains textToFind.

Usage

```
containsText(textToSearch, textToFind)
```

Arguments

`textToSearch`	The value to be searched.
`textToFind`	The value to find.

Value

TRUE if the textToFind value is found.

`convertPvtStyleToBasicStyle` *Convert a pivot table style to a basictabler style.*

Description

`convertPvtStyleToBasicStyle` is a utility function that converts a pivot table style to a basictabler style from the basictabler package.

Usage

```
convertPvtStyleToBasicStyle(btbl = NULL, pvtStyle = NULL)
```

Arguments

`btbl`	The basic table that will own the new style.
`pvtStyle`	The pivot style to convert.

Value

a basictabler style.

`convertPvtTblToBasicTbl` *Convert a pivot table to a basic table.*

Description

`convertPvtTblToBasicTbl` is a utility function that converts a pivot table to a basic table from the basictabler package.

Usage

```
convertPvtTblToBasicTbl(pvt = NULL, exportOptions = NULL,
  compatibility = NULL)
```

Arguments

`pvt`	The pivot table to convert.
`exportOptions`	Options specifying how values are exported.
`compatibility`	Compatibility options specified when creating the basictabler table.

Value

a basictabler table.

exportValueAs *Replace the current value with a placeholder during export.*

Description

exportValueAs is a utility function that returns either the original value or a replacement placeholder value for export.

Usage

```
exportValueAs(rawValue, formattedValue, exportOptions,
  blankValue = character(0))
```

Arguments

rawValue	The raw value to check.
formattedValue	The formatted value to be exported.
exportOptions	A list of options controlling export behaviour.
blankValue	The 'placeholder' value to be exported when skipping the value.

Value

Either the original value or a placeholder value.

getBlankTheme *Get an empty theme for applying no styling to a table.*

Description

Get an empty theme for applying no styling to a table.

Usage

```
getBlankTheme(parentPivot, themeName = "blank")
```

Arguments

parentPivot	Owning pivot table.
themeName	The name to use as the new theme name.

Value

A TableStyles object.

getCompactTheme *Get the compact theme for styling a pivot table.*

Description

Get the compact theme for styling a pivot table.

Usage

```
getCompactTheme(parentPivot, themeName = "compact")
```

Arguments

parentPivot	Owning pivot table.
themeName	The name to use as the new theme name.

Value

A PivotStyles object.

getDefaultTheme *Get the default theme for styling a pivot table.*

Description

Get the default theme for styling a pivot table.

Usage

```
getDefaultTheme(parentPivot, themeName = "default")
```

Arguments

parentPivot	Owning pivot table.
themeName	The name to use as the new theme name.

Value

A PivotStyles object.

getLargePlainTheme *Get the large plain theme for styling a pivot table.*

Description

Get the large plain theme for styling a pivot table.

Usage

```
getLargePlainTheme(parentPivot, themeName = "largeplain")
```

Arguments

parentPivot	Owning pivot table.
themeName	The name to use as the new theme name.

Value

A PivotStyles object.

getNextPosition *Find the first value in an array that is larger than the specified value.*

Description

getNextPosition is a utility function that helps when parsing strings that contain delimiters.

Usage

```
getNextPosition(positions, afterPosition)
```

Arguments

positions	An ordered numeric vector.
afterPosition	The value to start searching after.

Value

The first value in the array larger than afterPosition.

getPvtStyleDeclarations *Get pivot table style declarations from a pivot table style.*

Description

`getPvtStyleDeclarations` is a utility function that reads the styles from a pivot table style.

Usage

```
getPvtStyleDeclarations(pvtStyle = NULL)
```

Arguments

`pvtStyle`	The pivot table style to read.

Value

a list of style declarations.

getSimpleColoredTheme *Get a simple coloured theme.*

Description

Get a simple coloured theme that can be used to style a pivot table into a custom colour scheme.

Usage

```
getSimpleColoredTheme(parentPivot, themeName = "coloredTheme", colors,
  fontName)
```

Arguments

`parentPivot`	Owning pivot table.
`themeName`	The name to use as the new theme name.
`colors`	The set of colours to use when generating the theme (see the Styling vignette for details).
`fontName`	The name of the font to use, or a comma separated list (for font-fall-back).

Value

A PivotStyles object.

getTheme *Get a built-in theme for styling a pivot table.*

Description

getTheme returns the specified theme.

Usage

```
getTheme(parentPivot, themeName = NULL)
```

Arguments

parentPivot	Owning pivot table.
themeName	The name of the theme to retrieve.

Value

A PivotStyles object.

getXlBorderFromCssBorder *Convert CSS border values to those used by the openxlsx package.*

Description

getXlBorderFromCssBorder parses the CSS combined border declarations (i.e. border, border-left, border-right, border-top, border-bottom) and returns a list containing an openxlsx border style and color as separate elements.

Usage

```
getXlBorderFromCssBorder(text)
```

Arguments

text	The border declaration to parse.

Value

A list containing two elements: style and color.

getXlBorderStyleFromCssBorder *Convert CSS border values to those used by the openxlsx package.*

Description

`getXlBorderStyleFromCssBorder` takes border parameters expressed as a list (containing elements: width and style) and returns a border style that is compatible with the openxlsx package.

Usage

```
getXlBorderStyleFromCssBorder(border)
```

Arguments

`border`	A list containing elements width and style.

Value

An openxlsx border style.

isNumericValue *Check whether a numeric value is present.*

Description

`isNumericValue` is a utility function returns TRUE only when a numeric value is present. NULL, NA, numeric(0) and integer(0) all return FALSE.

Usage

```
isNumericValue(value)
```

Arguments

`value`	The value to check.

Value

TRUE if a numeric value is present.

isTextValue *Check whether a text value is present.*

Description

isTextValue is a utility function returns TRUE only when a text value is present. NULL, NA, character(0) and "" all return FALSE.

Usage

```
isTextValue(value)
```

Arguments

value	The value to check.

Value

TRUE if a non-blank text value is present.

oneToNULL *Convert a value of 1 to a NULL value.*

Description

oneToNull is a utility function that returns NULL when a value of 0 or 1 is passed to it, otherwise it returns the original value.

Usage

```
oneToNULL(value, convertOneToNULL)
```

Arguments

value	The value to check.
convertOneToNULL	TRUE to convert 1 to NULL.

Value

NULL if value==1, otherwise value.

parseColor *Convert a CSS colour into a hex based colour code.*

Description

parseColor converts a colour value specified in CSS to a hex based colour code. Example supported input values/formats/named colours are: #0080FF, rgb(0, 128, 255), rgba(0, 128, 255, 0.5) and red, green, etc.

Usage

```
parseColor(color)
```

Arguments

color	The colour to convert.

Value

The colour as a hex code, e.g. #FF00A0.

parseCssBorder *Parse a CSS border value.*

Description

parseCssBorder parses the CSS combined border declarations (i.e. border, border-left, border-right, border-top, border-bottom) and returns a list containing the width, style and color as separate elements.

Usage

```
parseCssBorder(text)
```

Arguments

text	The border declaration to parse.

Value

A list containing three elements: width, style and color.

parseCssSizeToPt *Convert a CSS size value into points.*

Description

parseCssSizeToPt will take a CSS style and convert it to points. Supported input size units are in, cm, mm, pt, pc, px, em, are converted exactly: in, cm, mm, pt, pc: using 1in = 2.54cm = 25.4mm = 72pt = 6pc. The following are converted approximately: px, em, approx 1em=16px=12pt and 100

Usage

```
parseCssSizeToPt(size)
```

Arguments

size	A size specified in common CSS units.

Value

The size converted to points.

parseCssSizeToPx *Convert a CSS size value into pixels*

Description

parseCssSizeToPx will take a CSS style and convert it to pixels Supported input size units are in, cm, mm, pt, pc, px, em, are converted exactly: in, cm, mm, pt, pc: using 1in = 2.54cm = 25.4mm = 72pt = 6pc. The following are converted approximately: px, em, approx 1em=16px=12pt and 100

Usage

```
parseCssSizeToPx(size)
```

Arguments

size	A size specified in common CSS units.

Value

The size converted to pixels.

parseCssString *Split a CSS attribute value into a vector/array.*

Description

parseCssString is a utility function that splits a string into a vector/array. The function pays attention to text qualifiers (single and double quotes) so won't split if the delimiter occurs inside a value.

Usage

```
parseCssString(text, separator = ",", removeEmptyString = TRUE)
```

Arguments

text	The text to split.
separator	The field separator, default comma.
removeEmptyString	TRUE to not return empty string / whitespace values.

Value

An R vector containing the values from text split up.

parseXlBorder *Parse an xl-border value.*

Description

parseXlBorder parses the combined xl border declarations (i.e. xl-border, xl-border-left, xl-border-right, xl-border-top, xl-border-bottom) and returns a list containing style and color as separate elements.

Usage

```
parseXlBorder(text)
```

Arguments

text	The border declaration to parse.

Value

A list containing two elements: style and color.

PivotBatch *A class that represents a batch calculation.*

Description

The PivotBatch class represents one combination of variables and calculations that are needed when calculating the values of cells in a pivot table.

Usage

```
PivotBatch
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that help to (do xyz).

Fields

`parentPivot` Owning pivot table.

`batchId` A unique integer identifier for this batch.

`compatibleCount` The number of pivot cell calculations that this batch supports.

`evaluated` TRUE if this batch has been evaluated.

`results` The results (a data frame) of the evaluation of this batch.

`asString` A text description of this batch.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new Pivot Batch.

`isCompatible(dataName=NULL, variableNames=NULL)` Checks whether the specified data name and combination of variable names are compatible with this batch.

`addCompatible(values=NULL, calculationName=NULL, calculationGroupName=NULL)` Adds another cell calculation to this batch.

`getCalculationInternalName(calculationName=NULL, calculationGroupName=NULL)` Find the internal name for this calculation.

`evaluateBatch()` Evaluate this batch.

`getSummaryValueFromBatch(filters=NULL, calculationName=NULL, calculationGroupName=NULL)` Retrieve a value from a batch that has already been evaluated.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotBatchCalculator` *A class that calculates the values for multiple cells.*

Description

The PivotBatchCalculator class calculates the values for multiple cells in the pivot table in one evaluation step.

Usage

```
PivotBatchCalculator
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that help to perform calculations in batch.

Fields

`parentPivot` Owning pivot table.

`batchCount` The number of batches generated for the pivot table.

`calculationSummary` A summary of the batch compatibility for each calculation.

`batchSummary` A summary of the batches in the pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new batch calculator.

`reset()` Clears any batches that currently exist in the batch calculator.

`isFiltersBatchCompatible(filters=NULL)` Determines whether a set of filters are compatible with batch calculations.

`generateBatchesForNamedCalculationEvaluation1(dataName=NULL, calculationName=NULL, calculationGroupN` Generates one or more batches for a named calculation and single working filters object.

`generateBatchesForNamedCalculationEvaluation2(calculationName=NULL, calculationGroupName=NULL, worki` Generates one or more batches for a named calculation and set of working working data associated with a cell.

`generateBatchesForCellEvaluation()` Generates one or batches for a pivot table cell.

`evaluateBatches()` Evaluates batch calculations.

`getSummaryValueFromBatch(batchName=NULL, calculationName=NULL, calculationGroupName=NULL, workingFilt`
Retrieve a value from a the results of a batch that has been evaluated.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotBatchStatistics` *A class that provides summary statistics for batch calculations.*

Description

The PivotBatchStatistics class contains a set of summary statistics that track how many calculations are batch compatible/incompatible.

Usage

```
PivotBatchStatistics
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that help to (do xyz).

Fields

`parentPivot` Owning pivot table.

`asString` A text description of the batch statistics.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new Pivot Batch Statistics.

`reset()` Clear the batch statistics.

`incrementNoData()` Increment the noData count for a batch.

`incrementCompatible()` Increment the compatible count for a batch.

`incrementIncompatible()` Increment the incompatible count for a batch.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

PivotCalculation *A class that defines a calculation.*

Description

The PivotCalculation class defines one calculation in a pivot table.

Usage

```
PivotCalculation
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that define a single pivot table calculation.

Fields

`parentPivot` Owning pivot table.

`calculationName` Calculation unique name.

`caption` Calculation display name - i.e. the name shown in the pivot table.

`visible` Show or hide the calculation. Hidden calculations are typically used as base values for other calculations.

`displayOrder` The order the calculations are displayed in the pivot table.

`filters` Any additional data filters specific to this calculation. This can be a PivotFilters object that further restrict the data for the calculation of a list of individual PivotFilter objects that provide more flexibility (and/or/replace). See the Calculations vignette for details.

`format` A character, list or custom function to format the calculation result.

`dataName` Specifies which data frame in the pivot table is used for this calculation.

`type` The calculation type: "summary", "calculation", "function" or "value".

`valueName` For type="value", the name of the column containing the value to display in the pivot table.

`summariseExpression` For type="summary", either the dplyr expression to use with dplyr::summarise() or a data.table calculation expression.

`calculationExpression` For type="calculation", an expression to combine aggregate values.

`calculationFunction` For type="function", a reference to a custom R function that will carry out the calculation.

`basedOn` A character vector specifying the names of one or more calculations that this calculation depends on.

`noDataValue` An integer or numeric value specifying the value to use if no data exists for a particular cell.

`noDataCaption` A character value that will be displayed by the pivot table if no data exists for a particular cell.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot calculation, specifying the field values documented above.

`asList()` Get a list representation of this calculation.

`asJSON()` Get a JSON representation of this calculation.

`asString()` Get a text representation of this calculation.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotCalculationGroup` *A class that defines a group of calculations.*

Description

The PivotCalculationGroup class is a container for multiple PivotCalculation objects.

Usage

```
PivotCalculationGroup
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that define a group of calculations.

Fields

`parentPivot` Owning pivot table.

`calculationGroupName` Calculation group unique name. Recommendation: Do not have spaces in this name.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot calculation group, specifying the field values documented above.

`isExistingCalculation(calculationName=NULL)` Check if a calculation exists with the specified name in this calculation group.

`getCalculation(calculationName=NULL)` Get the calculation with the specified name.

`defineCalculation(calculationName=NULL, caption=NULL, visible=TRUE, displayOrder=NULL, filters=NULL, f`
Create a new calculation. See the `PivotCalculation` class documentation for more details on the arguments.

`asList()` Get a list representation of this calculation group.

`asJSON()` Get a JSON representation of this calculation group.

`asString()` Get a text representation of this calculation group.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

PivotCalculationGroups

A class that contains multiple calculation groups.

Description

The PivotCalculationGroups class stores all of the calculation groups for a pivot table.

Usage

```
PivotCalculationGroups
```

Format

`R6Class` object.

Value

Object of `R6Class`.

Fields

`parentPivot` Owning pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new container for pivot table calculation groups, specifying the field values documented above.

`isExistingCalculationGroup(calculationGroupName)` Check if a calculation group exists with the specified name.

`getCalculationGroup(calculationGroupName)` Get the calculation group with the specified name.

`addCalculationGroup(calculationGroupName)` Create a new calculation group with the specified name.

`asList()` Get a list representation of these calculation groups.

`asJSON()` Get a JSON representation of these calculation groups.

`asString()` Get a text representation of these calculation groups.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotCalculator` *A class that computes the value of a cell.*

Description

The PivotCalculator class has various functions and methods that assist with calculating the value of a cell in a pivot table.

Usage

```
PivotCalculator
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that help calculate the value of a pivot table cell.

Fields

`parentPivot` Owning pivot table.

`batchInfo` Get a summary of the batch calculations.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot table calculator, specifying the field value documented above.

`getDataFrame(dataName)` Gets a data frame with the specified name from the data frames added to the pivot table.

`getCalculationGroup(calculationGroupName)` Gets a calculation group with the specified name from the calculation groups added to the pivot table.

`getCalculation(calculationGroupName, calculationName)` Gets a calculation with the specified name and group from the calculation groups added to the pivot table.

`generateBatchesForCellEvaluation()` Examines the cells in the pivot table to generate one or more batch calculations.

`evaluateBatches()` Evaluate batch calculations using the batch calculator.

`newFilter(variableName, values)` Creates a new PivotFilter object associated with the specified data frame column name and column values.

`newFilters(variableName, values)` Creates a new PivotFilters object associated with the specified data frame column name and column values.

`setFilters(filters1, filters2, action="replace")` Combines two PivotFilters objects (e.g. to intersect the filters coming from the row and column headings for a particular cell).

`setFilterValues(filters, variableName, values, action="replace")` Updates a PivotFilters object based on a PivotFilter object (e.g. to union the filter criteria arising from multiple row headers).

`getFilteredDataFrame(dataFrame, filters)` Applies a PivotFilters object to a data frame, returning a new data frame.

`getDistinctValues(dataFrame, variableName)` Gets the distinct values from the specified column of a data frame.

`formatValue(value, format)` Formats a numerical value using either an sprintf string, a list of arguments for the base::format() function or using a custom R function.

`getCombinedFilters = function(rowColFilters=NULL, calcFilters=NULL, cell=NULL)` Get the working filters for a calculation.

`getFiltersForNamedCalculation = function(calculationName=NULL, calculationGroupName=NULL, rowColFilte` Get the working filters for a named calculation.

`setWorkingData = function(cell=NULL)` Set the working filters for a cell.

`evaluateSingleValue(dataFrame, workingFilters, valueName, format, noDataValue, noDataCaption)` Get a single value from a data frame.

`evaluateSummariseExpression(dataName=NULL, dataFrame=NULL, workingFilters=NULL, calculationName=NULL,` Calculate a summary value, either using a batch or sequential calculation.

`evaluateCalculationExpression(values, calculationExpression, format, noDataValue, noDataCaption)` Evaluates an R expression in order to combine the results of other calculations.

`evaluateCalculateFunction(workingFilters, calculationFunction, format, baseValues, cell)` Invokes a user-provided custom R function to aggregate data and perform calculations.

`evaluateNamedCalculationWD(calculationName, calculationGroupName, workingData, cell)` Invokes the relevant calculation function based upon the calculation type.

`evaluateNamedCalculation(calculationName, calculationGroupName, rowColFilters)` Determines the working filters and invokes the relevant calculation function based upon the calculation type.

`evaluateCell(cell)` Top-level calculation function responsible for calculating the value of a pivot table cell.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotCell` *A class that represents a cell in a pivot table*

Description

The PivotCell class represents a cell in the body of a pivot table (i.e. not a row/column heading, rather a cell typically containing a numerical value).

Usage

```
PivotCell
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that define a single pivot table cell

Fields

`parentPivot` Owning pivot table.

`rowNumber` The row number of the cell. 1 = the first (i.e. top) data row.

`columnNumber` The column number of the cell. 1 = the first (i.e. leftmost) data column.

`rowLeafGroup` The row data group linked to this row.

`columnLeafGroup` The column data group linked to this column.

`calculationName` The name of the calculation that is displayed in the cell.

`calculationGroupName` The name of the calculation group that owns the above calculation.

`rowFilters` The data filters applied to this cell from the row headings.

`columnFilters` The data filters applied to this cell from the column headings.

`rowColFilters` The data filters applied to this cell from both the row and column headings.

`calculationFilters` The data filters applied to this cell from the calculation definition.

`workingData` The data filters and batchNames used applied when running calculations (including the filters needed for base calculations when calculation type="calculation").

`evaluationFilters` The final and actual data filters used in the calculation of the cell value (e.g. custom calculation functions can override the working filters).

`isTotal` Whether this cell is a total cell.

`rawValue` The numerical calculation result.

`formattedValue` The formatted calculation result (i.e. character data type).

`baseStyleName` The name of the style applied to this cell (a character value). The style must exist in the PivotStyles object associated with the PivotTable.

`style` A PivotStyle object that can apply overrides to the base style for this cell.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot table cell, specifying the field values documented above.

`getCopy())` Get a copy of this cell.

`asList())` Get a list representation of this cell

`asJSON()` Get a JSON representation of this cell

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

PivotCells *A class that contains the cells from a pivot table.*

Description

The PivotCells class contains all of the PivotCell objects that comprise the body of a pivot table.

Usage

```
PivotCells
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods relating to the cells of a pivot table.

Fields

`parentPivot` Owning pivot table.

`rowGroups` The row data groups that represent the row headings in the pivot table.

`columnGroups` The column data groups that represent the column headings in the pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new set of pivot table cells, specifying the field values documented above.

`getCell(r, c))` Get the PivotCell at the specified row and column coordinates in the pivot table.

`setCell(r, c, cell))` Set the PivotCell at the specified row and column coordinates in the pivot table.

`getCells(specifyCellsAsList=FALSE, rowNumbers=NULL, columnNumbers=NULL, cellCoordinates=NULL)` Retrieve cells by a combination of row and/or column numbers.

`findCells(variableNames=NULL, variableValues=NULL, totals="include", calculationNames=NULL, minValue=` Find cells matching the specified criteria.

`getColumnWidths())` Retrieve the width of the longest value (in characters) in each column.

`asMatrix(rawValue=TRUE))` Get a matrix containing all of the numerical values from the body of the pivot table (for rawValue=TRUE) or all of the formatted (i.e. character) values (for rawValue=FALSE).

`asList())` Get a list representation of the pivot table cells.

`asJSON()` Get a JSON representation of the pivot table cells.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotData` *A class that contains named data frames.*

Description

The PivotData class stores all of the data frames associated with a pivot table.

Usage

```
PivotData
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that help quickly storing and retrieving data frames.

Fields

`parentPivot` Owning pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot data container, specifying the field value documented above.

`addData(df, dataName)` And a data frame to the pivot table, specifying a name that can be used to easily retrieve it or refer to it later.

`getData(dataName)` Get the data frame with the specified name.

`isKnownData(dataName))` Check if a data frame exists with the specified name.

`asList()` Get a list representation of the contained data frames.

`asJSON()` Get a JSON representation of the contained data frames.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

PivotDataGroup *A class that defines a row or column heading.*

Description

The PivotDataGroup class represents one row or column heading in a pivot table. PivotDataGroups have a parent-child relationship, i.e. each PivotDataGroup can have one or more child PivotDataGroups.

Usage

```
PivotDataGroup
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that help define the row and column headings in a pivot table.

Fields

`parentGroup` Parent PivotDataGroup.

`parentPivot` Owning pivot table.

`rowOrColumn` "row" or "column" indicating which axis this data group exists on.

`caption` The friendly display name for this data group.

`variableName` The name of the related column in the data frame(s) of the pivot table.

`values` The data value(s) which this data group represents. Can be a vector of values.

`sortValue` The data value used when sorting the data groups.

`isTotal` Whether this data group is a total group.

`isLevelSubTotal` Whether this data group is a sub-total group in the current level.

`isLevelTotal` Whether this data group is a total group in the current level.

`visualTotals` Whether visual totals are enabled for this data group.

`calculationGroupName` The name of the calculation group applied to this data group.

`calculationName` The name of the calculation applied to this data group.

`rowColumnNumber` The row or column number of this data group, i.e. where it exists in the pivot table.

`baseStyleName` The name of the style applied to this data group (i.e. this row/column heading). The style must exist in the PivotStyles object associated with the PivotTable.

`fixedWidthSize` the width of this data group in characters. Only applies to column data groups, not row data groups.

`style` A PivotStyle object that can apply overrides to the base style for this data group.

`isMatch` Whether or not this data group matches the criteria of the last find.

`isRendered` Whether or not this data group has been rendered yet (used as part of the rendering routines).

`isWithinVisibleRange` whether or not this data group is visible (used as part of the rendering routines).

`visibleChildGroupCount` The number of visible child groups (used as part of the rendering routines)

`visibleDescendantGroupCount` The number of visible descendant groups (used as part of the rendering routines).

`visibleLeafGroupCount` The number of visible leaf groups beneath this group (used as part of the rendering routines).

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot data group, specifying the field values documented above.

`getLevelNumber()` Get the level number of this data group, where level 1 is the top data group.

`getAncestorGroups(ancestors, includeCurrentGroup=FALSE)` Get all of the data groups above the current data group in the parent-child data group hierarchy.

`getDescendantGroups(descendants, includeCurrentGroup=FALSE)` Get all of the data groups below the current data group in the parent-child data group hierarchy.

`getLeafGroups(leafGroups)` Get all of the data groups across the bottom level of the data group hierarchy.

`getLevelCount(includeCurrentLevel=FALSE)` Count the number of levels in the data group hierarchy.

`getLevelGroups(level, levelGroups)` Get all of the data groups at a specific level in the data group hierarchy.

`addChildGroup(variableName, values, caption, isTotal=FALSE, isLevelSubTotal=FALSE, isLevelTotal=FALSE,` Add a new data group as the child of the current data group.

`addDataGroups(variableName, atLevel, fromData=TRUE, dataName, dataSortOrder="asc", dataFormat, onlyCom` Generate new data groups based on the distinct values in a data frame or using explicitly specified data values.

`sortDataGroups(levelNumber=1, orderBy="calculation", sortOrder="desc", calculationGroupName="default'` Sort data groups either by the data group data value, caption or based on calculation result values.

`addCalculationGroups(calculationGroupName, atLevel)` Add a calculation group to the data group hierarchy.

`normaliseDataGroup()` Normalise the data group hierarchy so that all branches have the same number of levels - accomplished by adding empty child data groups where needed.

`getNetFilters()` Get a PivotFilters object that contains all of the filters applied in this data group and all of its ancestors.

`getNetCalculationName()` Get the calculation name applied in this data group or its nearest ancestor.

`isFindMatch(matchMode="simple", variableNames=NULL, variableValues=NULL, totals="include", calculatio` Tests whether this data group matches the specified criteria.

`findDataGroups(matchMode="simple", variableNames=NULL, variableValues=NULL, totals="include", calcula` Searches all data groups underneath this data group to find groups that match the specified criteria.

`asList()` Get a list representation of the data group(s).

`asJSON()` Get a JSON representation of the data group(s).

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotFilter` *A class that defines a filter condition.*

Description

The PivotFilter class represents a single filter condition. The condition relates to one column and is of the form [ColumnName] IN c(Value1, Value2, Value3, ...). Often in a pivot table, each filter specifies only one data value, as typically each distinct data value exists in a separate row or column.

Usage

```
PivotFilter
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that define a single pivot table filter.

Fields

`parentPivot` Owning pivot table.

`variableName` The name of the column in the data frame that this filter will apply to.

`safeVariableName` The name of the column, surrounded by back-ticks, if the name is not legal.

`values` A single data value or a vector of data values that could/can be found in the data frame column.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot filter, specifying the field values documented above.

`intersect(filter)` Update this PivotFilter by intersecting the allowed values in this filter with the allowed values in the specified filter.

`union(filter)` Update this PivotFilter by unioning the allowed values in this filter with the allowed values in the specified filter.

`replace(filter)` Update this PivotFilter by replacing the allowed values in this filter with the allowed values from the specified filter.

`getCopy()` Get a copy of this PivotFilter.

`asList()` Get a list representation of this PivotFilter.

`asJSON()` Get a list representation of this PivotFilter.

`asString(includeVariableName=TRUE, seperator=" ")` Get a text representation of this PivotFilter.

Examples

```
pt <- PivotTable$new()
# ...
PivotFilter$new(pt, variableName="Country", values="England")
```

PivotFilterOverrides *A class that defines a set of filter overrides*

Description

The PivotFilterOverrides class contains multiple PivotFilter objects that can be used later to override a set of filters, e.g. in a pivot table calculation.

Usage

```
PivotFilterOverrides
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that define a set of filters and associated override actions

Fields

`parentPivot` Owning pivot table.

`removeAllFilters` TRUE to remove all existing filters before applying any other and/replace/or filters.

`keepOnlyFiltersFor` Specify the names of existing variables to retain the filters for. All other filters will be removed.

`removeFiltersFor` Specify the names of variables to remove filters for.

`overrideFunction` A custom function to amend the filters in each cell.

`countIntersect` The number of PivotFilters that will be combined with other pivot filters by intersecting their lists of allowed values.

`countReplace` The number of PivotFilters that will be combined with other pivot filters by entirely replacing existing PivotFilter objects.

`countUnion` The number of PivotFilters that will be combined with other pivot filters by unioning their lists of allowed values.

`countTotal` The total number of PivotFilters that will be combined with other pivot filters.

`intersectFilters` The PivotFilters that will be combined with other pivot filters by intersecting their lists of allowed values.

`replaceFilters` The PivotFilters that will be combined with other pivot filters by entirely replacing existing PivotFilter objects.

`unionFilters` The PivotFilters that will be combined with other pivot filters by unioning their lists of allowed values.

`allFilters` The complete set of PivotFilters that will be combined with other pivot filters.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot filter overrides object, specifying the field values documented above.

`add(filter=NULL, variableName=NULL, type="ALL", values=NULL, action="intersect")` Add a pivot filter override, either from an existing PivotFilter object or by specifying a variableName and values.

`apply(filters)` Apply the filter overrides to a PivotFilters object.

`asList()` Get a list representation of this PivotFilterOverrides object.

`asJSON()` Get a JSON representation of this PivotFilterOverrides object.

`asString(includeVariableName=TRUE, seperator=", ")` Get a text representation of this PivotFilterOverrides object.

Examples

```
pt <- PivotTable$new()
# ...
# PivotFilterOverrides constructor allows a filter to be defined
# e.g. to enable %of row or column type calculations
filterOverrides <- PivotFilterOverrides$new(pt, keepOnlyFiltersFor="Volume")
# Alternatively/in addition, create a new filter
filter <- PivotFilter$new(pt, variableName="Country", values="England")
# Add the filter to the set of overrides
filterOverrides$add(filter=filter, action="replace")
```

PivotFilters *A class that defines a set of filter conditions*

Description

The PivotFilters class allows multiple filter conditions relating to different data frame columns to be combined, i.e. a PivotFilters object can contain multiple `PivotFilter` objects.

Usage

```
PivotFilters
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that define a set of filter conditions.

Fields

`parentPivot` Owning pivot table.

`count` The number of PivotFilter objects in this PivotFilters object.

`filters` The PivotFilter objects in this PivotFilters object.

`isALL` If TRUE, this PivotFilters object matches all data.

`isNONE` If TRUE, this PivotFilters object matches no data.

`filteredVariables` A list of the variables that are filtered by this PivotFilters object.

`filteredValues` A list of the criteria values for each of the variables filtered by this PivotFilters object.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot filters object, specifying the field values documented above.

`clearFilters()` Remove all filters.

`getFilter(variableName=NULL)` Find a filter with the specified variable name.

`isFilterMatch(variableNames=NULL, variableValues=NULL)` Test whether these filters match the specified criteria.

`setFilters(filters=NULL, action="replace")` Update the value of this PivotFilters object with the filters from the specified PivotFilters object, either unioning, intersecting or replacing the filter criteria.

`setFilter(filter=NULL, action="replace")` Update the value of this PivotFilters object with the specified PivotFilter object, either unioning, intersecting or replacing the filter criteria.

`setFilterValues(variableName=NULL, type=NULL, values=NULL, action="replace")` Update the value of this PivotFilters object with the specified criteria, either unioning, intersecting or replacing the filter criteria.

`addFilter()` Directly add a PivotFilter object to this PivotFilters object.

`getFilteredDataFrame(dataFrame=NULL)` Filters the specified data frame and returns the results as another data frame.

`getCopy()` Get a copy of this set of filters.

`asList()` Get a list representation of this PivotFilters object.

`asJSON()` Get a JSON representation of this PivotFilters object.

`asString(includeVariableName=TRUE, seperator=", ")` Get a text representation of this PivotFilters object.

Examples

```
pt <- PivotTable$new()
# ...
# PivotFilters constructor allows a filter to be defined
filters <- PivotFilters$new(pt, variableName="Year", values=2017)
# Create a new filter
```

```
filter <- PivotFilter$new(pt, variableName="Country", values="England")
# Combine the filters
filters$setFilter(filter)
# filters now contains criteria for both Year and Country
# Now add another filter, this time via an alternative method
filters$setFilterValues(variableName="Product", values="Cadbury Dairy Milk
Chocolate 100g")
# filters now contains criteria for Year, Country and Product
```

`PivotHtmlRenderer` *A class that renders a pivot table in HTML.*

Description

The PivotHtmlRenderer class creates a HTML representation of a pivot table.

Usage

```
PivotHtmlRenderer
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that render to HTML.

Fields

`parentPivot` Owning pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot table renderer, specifying the field value documented above.

`clearIsRenderedFlags()` Clear the IsRendered flags that exist on the PivotDataGroup class.

`getTableHtml(styleNamePrefix=NULL, includeHeaderValues=FALSE, includeRCFilters=FALSE, includeCalculat` Get a HTML representation of the pivot table, optionally including additional detail for debugging purposes.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

PivotLatexRenderer *A class that renders a pivot table in Latex.*

Description

The PivotLatexRenderer class creates a Latex representation of a pivot table.

Usage

```
PivotLatexRenderer
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that render to Latex.

Fields

`parentPivot` Owning pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot table renderer, specifying the field value documented above.

`clearIsRenderedFlags()` Clear the IsRendered flags that exist on the PivotDataGroup class.

`resetVisibleRange()` Clears the visible range that has been set, so the next call to getTableLatex() will render the whole table.

`setVisibleRange(fromRow=NULL, toRow=NULL, fromColumn=NULL, toColumn=NULL)` Specifies a subset of the pivot table to be rendered, e.g. for use when a pivot table will not fit into a single A4 page.

`clearIsRenderedFlags()`

`getTableLatex = function(caption=NULL, label=NULL, boldHeadings=FALSE, italicHeadings=FALSE, exportOpt` Get a Latex representation of the pivot table, specifying the caption to appear above the table, the label to use when referring to the table elsewhere in the document and how headings should be styled.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotOpenXlsxRenderer` *A class that renders a pivot table into an Excel worksheet.*

Description

The PivotOpenXlsxRenderer class creates a representation of a pivot table in an Excel file using the openxlsx package.

Usage

```
PivotOpenXlsxRenderer
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that render to Excel via the openxlsx package

Fields

`parentPivot` Owning pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new pivot table renderer, specifying the field value documented above.

`clearIsRenderedFlags()` Clear the IsRendered flags that exist on the PivotDataGroup class.

`writeToCell(wb=NULL, wsName=NULL, rowNumber=NULL, columnNumber=NULL, value=NULL, applyStyles=TRUE, bas`
Writes a value to a cell and applies styling as needed.

`writeToWorksheet(wb=NULL, wsName=NULL, topRowNumber=NULL, leftMostColumnNumber=NULL, outputValuesAs="`
Output the pivot table into the specified workbook and worksheet at the specified row-column location.

Examples

```
# This class should only be created by the pivot table.
# It is not intended to be created outside of the pivot table.
```

`PivotOpenXlsxStyle` *A class that specifies Excel styling as used by the openxlsx package.*

Description

The PivotOpenXlsxStyle class specifies the styling for cells in an Excel worksheet.

Usage

```
PivotOpenXlsxStyle
```

Format

`R6Class` object.

Value

Object of `R6Class` with properties and methods that help define styles.

Fields

`parentPivot` Owning pivot table.

`baseStyleName` The name of the base style in the pivot table.

`isBaseStyle` TRUE when this style is the equivalent of a named style in the pivot table, FALSE if this style has additional settings over and above the base style of the same name.

`fontName` The name of the font (single font name, not a CSS style list).

`fontSize` The size of the font (units: point).

`bold` TRUE if text is bold.

`italic` TRUE if text is italic.

`underline` TRUE if text is underlined.

`strikethrough` TRUE if text has a line through it.

`superscript` TRUE if text is small and raised.

`subscript` TRUE if text is small and lowered.

`fillColor` The background colour for the cell (as a hex value, e.g. #00FF00).

`textColor` The color of the text (as a hex value).

`hAlign` The horizontal alignment of the text: left, center or right.

`vAlign` The vertical alignment of the text: top, middle or bottom.

`wrapText` TRUE if the text is allowed to wrap onto multiple lines.

`textRotation` The rotation angle of the text or 255 for vertical.

`border` A list (with elements style and color) specifying the border settings for all four sides of each cell at once.

`borderLeft` A list (with elements style and color) specifying the border settings for the left border of each cell.

`borderRight` A list (with elements style and color) specifying the border settings for the right border of each cell.

`borderTop` A list (with elements style and color) specifying the border settings for the top border of each cell.

`borderBottom` A list (with elements style and color) specifying the border settings for the bottom border of each cell.

`valueFormat` The Excel formatting applied to the field value. One of the following values: GENERAL, NUMBER, CURRENCY, ACCOUNTING, DATE, LONGDATE, TIME, PERCENTAGE, FRACTION, SCIENTIFIC, TEXT, COMMA. Or for dates/datetimes, a combination of d, m, y. Or for numeric values, use 0.00 etc.

`minColumnWidth` The minimum width of this column.

`minRowHeight` The minimum height of this row.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new Excel style, specifying the field values documented above.

`isBasicStyleNameMatch(baseStyleName=NULL)` Find a matching base style by name.

`isFullStyleDetailMatch = function(baseStyleName=NULL, isBaseStyle=NULL, fontName=NULL, fontSize=NULL,` Find a matching style matching on all the attributes of the style.

`createOpenXslxStyle()` Create the openxlsx style.

`asList()` Get a list representation of this style.

`asJSON()` Get a JSON representation of this style.

`asString()` Get a text representation of this style.

Examples

```
# PivotOpenXlsxStyle objects are created by the PivotOpenXlsxRenderer class.
# See that class for details.
```

`PivotOpenXlsxStyles` *A class that defines a collection of Excel styles as used by the openxlsx package.*

Description

The PivotOpenXlsxStyles class stores a collection of PivotTableOpenXlsx style objects.

Usage

```
PivotOpenXlsxStyles
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that define styles/a theme for a pivot table.

Fields

`parentPivot` Owning pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new set of styles, specifying the field values documented above.

`findNamedStyle(baseStyleName)` Find an existing openxlsx style matching the name of a base style.

`findOrAddStyle(action="findOrAdd", baseStyleName=NULL, isBaseStyle=NULL, style=NULL, mapFromCss=TRUE)` Find an existing openxlsx style and/or add a new openxlsx style matching a base style and/or PivotStyle object.

`addNamedStyles(mapFromCss=TRUE)` Populate the OpenXlsx styles based on the styles defined in the pivot table.

`asList()` Get a list representation of the styles.

`asJSON()` Get a JSON representation of the styles.

`asString()` Get a text representation of the styles.

Examples

```
# This class is used internally by the Pivot Table.
```

PivotStyle *A class that specifies styling.*

Description

The PivotStyle class specifies the styling for headers and cells in a pivot table. Styles are specified in the form of Cascading Style Sheet (CSS) name-value pairs.

Usage

```
PivotStyle
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that help define styles.

Fields

`parentPivot` Owning pivot table.

`styleName` Style unique name.

`declarations` CSS style declarations.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new style declaration, specifying the field values documented above.

`setPropertyValue(property, value)` Set a single style property.

`setPropertyValues(declarations)` Set multiple style properties, where declarations is a list similar to the code example below.

`getPropertyValue(property)` Get the style declarations for a single property.

`asCSSRule(selector)` Get this style definition in the form of a CSS rule.

`asNamedCSSStyle(styleNamePrefix)` Get this style definition in the form of a named CSS style.

`getCopy(newStyleName)` Get a copy of this style.

`asList()` Get a list representation of this style.

`asJSON()` Get a JSON representation of this style.

Examples

```
# PivotStyle objects are normally created indirectly via one of the helper
# methods.
# For an example, see the PivotStyles class.
```

PivotStyles *A class that defines a collection of styles.*

Description

The PivotStyles class defines all of the base styles needed to style/theme a pivot table. It also defines the names of the styles that are used for styling the different parts of the pivot table.

Usage

```
PivotStyles
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that define styles/a theme for a pivot table.

Fields

`parentPivot` Owning pivot table.

`themeName` The name of the theme.

`allowExternalStyles` Enables integration scenarios where an external system is supplying the CSS definitions.

`tableStyle` The name of the style for the HTML table element.

`rootStyle` The name of the style for the HTML cell at the top left of the pivot table.

`rowHeaderStyle` The name of the style for the row headers in the pivot table.

`colHeaderStyle` The name of the style for the column headers in the pivot table.

`cellStyle` The name of the cell style for the non-total cells in the body of the pivot table.

`totalStyle` The name of the cell style for the total cells in the pivot table.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(...)` Create a new set of styles, specifying the field values documented above.

`isExistingStyle(styleName)` Check whether the specified style exists.

`getStyle(styleName)` Get the specified style.

`addStyle(styleName, declarations)` Add a new style to the collection of styles.

`copyStyle(styleName, newStyleName)` Create a copy of a style with the specified name.

`asCSSRule(styleName, selector)` Get a style definition in the form of a CSS rule.

`asNamedCSSStyle(styleName, styleNamePrefix)` Get a style definition in the form of a named CSS style.

`asList()` Get a list representation of the styles.

`asJSON()` Get a JSON representation of the styles.

`asString()` Get a text representation of the styles.

Examples

```
pt <- PivotTable$new()
# ...
pivotStyles <- PivotStyles$new(pt, themeName="compact")
pivotStyles$addStyle(styleName="MyNewStyle", list(
    font="0.75em arial",
    padding="2px",
    border="1px solid lightgray",
    "vertical-align"="middle",
    "text-align"="center",
    "font-weight"="bold",
    "background-color"="#F2F2F2"
  ))
```

PivotTable *A class that defines a pivot table.*

Description

The PivotTable class represents a pivot table and is the primary class for constructing and interacting with the pivot table.

Usage

```
PivotTable
```

Format

R6Class object.

Value

Object of R6Class with properties and methods that define a pivot table.

Fields

`argumentCheckMode` A number (0-4 meaning none, minimal, basic, balanced, full) indicating the argument checking level.

`traceEnabled` A logical value indicating whether actions are logged to a trace file.

`processingLibrary` A character value indicating the processing library being used (base, dplyr, data.table).

`data` A PivotData object containing the data frames used to populate the pivot table.

`rowGroup` The top PivotDataGroup in the parent-child hierarchy of row data groups.

`columnGroup` The top PivotDataGroup in the parent-child hierarchy of column data groups.

`calculationGroups` A PivotCalculationGroups object containing all of the pivot calculations in the pivot table.

`calculationsPosition` "row" or "column" indicating where the calculation names will appear (only if multiple calculations are defined and visible in the pivot table).

`evaluationMode` Either "sequential" or "batch" to specify how summary calculations (i.e. where type="summary") are evaluated.

`batchInfo` Get a text summary of the batch calculations from the last evaluation of this pivot table.

`cells` A PivotCells object containing all of the cells in the body of the pivot table.

`rowCount` The number of rows in the table.

`columnCount` The number of columns in the table.

`theme` The name of the theme currently applied to the pivot table.

`styles` A PivotStyles object containing the styles used to theme the pivot table.

`allowExternalStyles` Enable support for external styles, when producing content for external systems.

`allTimings` The time taken for various activities related to constructing the pivot table.

`significantTimings` The time taken for various activities related to constructing the pivot table, where the elapsed time > 0.1 seconds.

Methods

Documentation For more complete explanations and examples please see the extensive vignettes supplied with this package.

`new(processingLibrary="auto", evaluationMode="batch", argumentCheckMode="auto", theme=NULL, replaceEx` Create a new pivot table, optionally specifying styling and enabling debug logging.

`addData(df, dataName)` Add a data frame with the specified name to the pivot table.

`getTopColumnGroups()` Get the very top column PivotDataGroup that sits at the top of the parent-child hierarchy.

`getLeafColumnGroups()` Get the PivotDataGroups at the bottom of the column heading parent-child hierarchy.

`addColumnDataGroups(variableName, atLevel, fromData=TRUE, dataName, dataSortOrder="asc", dataFormat, o` Generate new column heading data groups based on the distinct values in a data frame or using explicitly specified data values.

`normaliseColumnGroups()` Normalise the column heading data group hierarchy so that all branches have the same number of levels - accomplished by adding empty child data groups where needed.

`sortColumnDataGroups(levelNumber=1, orderBy="calculation", sortOrder="desc", calculationGroupName="de` Sort the column heading data groups either by the data group data value, caption or based on calculation result values.

`getTopRowGroups()` Get the left-most row PivotDataGroup that sits at the top of the parent-child hierarchy.

`getLeafRowGroups()` Get the PivotDataGroups at the bottom of the row heading parent-child hierarchy.

`addRowDataGroups(variableName, atLevel, fromData=TRUE, dataName, dataSortOrder="asc", dataFormat, only` Generate new row heading data groups based on the distinct values in a data frame or using explicitly specified data values.

`normaliseRowGroups()` Normalise the row heading data group hierarchy so that all branches have the same number of levels - accomplished by adding empty child data groups where needed.

`sortRowDataGroups(levelNumber=1, orderBy="calculation", sortOrder="desc", calculationGroupName="defau` Sort the row heading data groups either by the data group data value, caption or based on calculation result values.

`addCalculationGroup(calculationGroupName)` Create a new calculation group (rarely needed since the default group is sufficient for almost all scenarios).

`defineCalculation(calculationGroupName="default", calculationName, caption, visible=TRUE, displayOrde` Define a new calculation. See the PivotCalculation class for details.

`addColumnCalculationGroups(calculationGroupName="default", atLevel, baseStyleName=NULL, styleDeclarat`
Add calculation names on columns (if more than one calculation is defined and visible, then the calculation names will appear as column headings).

`addRowCalculationGroups(calculationGroupName="default", atLevel, baseStyleName=NULL, styleDeclaration`
Add calculation names on rows (if more than one calculation is defined and visible, then the calculation names will appear as row headings).

`addStyle(styleName, declarations)` Define a new PivotStyle and add it to the PivotStyles collection.

`createInlineStyle(baseStyleName, declarations)` Create a PivotStyle object that can be used to style individual cell in the pivot table.

`setStyling(rFrom=NULL, cFrom=NULL, rTo=NULL, cTo=NULL, groups=NULL, cells=NULL, baseStyleName=NULL, sty`
Set the styling for a one/multiple data groups and/or cells in the pivot table.

`generateCellStructure()` Generate the empty pivot table cells (after the row/column headings have been defined).

`resetCells()` Clear the cells of the pivot table (should be done automatically after structural changes have been made to the pivot table).

`evaluateCells()` Calculate the values of the cells in the body of the pivot table.

`evaluatePivot()` A wrapper for calling normaliseColumnGroups(), normaliseRowGroups(), generateCellStructure() and evaluateCells() in sequence.

`findRowDataGroups(matchMode="simple", variableNames=NULL, variableValues=NULL, totals="include", calc`
Find row data groups matching the specified criteria.

`findColumnDataGroups(matchMode="simple", variableNames=NULL, variableValues=NULL, totals="include", c`
Find column data groups matching the specified criteria.

`getCells(specifyCellsAsList=FALSE, rowNumbers=NULL, columnNumbers=NULL, cellCoordinates=NULL)`
Retrieve cells by a combination of row and/or column numbers.

`findCells(variableNames=NULL, variableValues=NULL, totals="include", calculationNames=NULL, minValue=`
Find cells in the body of the pivot table matching the specified criteria.

`print(asCharacter=FALSE)` Either print the pivot table to the console or retrieve it as a character value.

`asMatrix(includeHeaders=TRUE, repeatHeaders=FALSE, rawValue=FALSE)` Gets the pivot table as a matrix, with or without headings.

`asDataFrame(separator=" ")` Gets the pivot table as a data frame, combining multiple levels of headings with the specified separator.

`asTidyDataFrame(includeGroupCaptions=TRUE, includeGroupValues=TRUE, separator=" ")`
Gets the pivot table as a tidy data frame, where each cell in the body of the pivot table becomes one row in the data frame.

`getCss(styleNamePrefix)` Get the CSS declarations for the entire pivot table.

`getHtml(styleNamePrefix, includeHeaderValues=FALSE, includeRCFilters=FALSE, includeCalculationFilters`
Get the HTML representation of the pivot table, specifying the CSS style name prefix to use and whether additional debug information should be included in the pivot table.

`saveHtml(filePath, fullPageHTML=TRUE, styleNamePrefix, includeHeaderValues=FALSE, includeRCFilters=FA`
Save the HTML representation of the pivot table to a file.

`renderPivot(width, height, styleNamePrefix, includeHeaderValues=FALSE, includeRCFilters=FALSE, include`
Render the pivot table as a htmlwidget.

`getLatex(caption=NULL, label=NULL, fromRow=NULL, toRow=NULL, fromColumn=NULL, toColumn=NULL, boldHeadi`
Get the Latex representation of the pivot table, specifying the caption to appear above the table, the label to use when referring to the table elsewhere in the document and how headings should be styled.

`writeToExcelWorksheet(wb=NULL, wsName=NULL, topRowNumber=NULL, leftMostColumnNumber=NULL, outputHeadi`
Output the pivot table into the specified workbook and worksheet at the specified row-column location.

`showBatchInfo()` Show a text summary of the batch calculations from the last evaluation of this pivot table.

`asList()` Get a list representation of the pivot table.

`asJSON()` Get a JSON representation of the pivot table.

`viewJSON()` View the JSON representation of the pivot table.

Examples

```
# The package vignettes have many more examples of working with the
# PivotTable class.
library(pivottabler)
pt <- PivotTable$new()
pt$addData(bhmtrains)
pt$addColumnDataGroups("TrainCategory")
pt$addRowDataGroups("TOC")
pt$defineCalculation(calculationName="TotalTrains",
summariseExpression="n()")
pt$renderPivot()
```

---

pivottabler *Render a pivot table as a HTML widget.*

Description

The `pivottabler` function is primarily intended for use with Shiny web applications.

Usage

```
pivottabler(pt, width = NULL, height = NULL, styleNamePrefix = NULL,
  includeRCFilters = FALSE, includeCalculationFilters = FALSE,
  includeWorkingData = FALSE, includeEvaluationFilters = FALSE,
  includeCalculationNames = FALSE, includeRawValue = FALSE)
```

Arguments

pt	The pivot table to render.
width	The target width.
height	The target height.
styleNamePrefix	A text prefix to be prepended to the CSS declarations (to ensure uniqueness).
includeRCFilters	Show/hide filter detail for debugging.
includeCalculationFilters	Show/hide filter detail for debugging.
includeWorkingData	Show/hide working data detail for debugging.
includeEvaluationFilters	Show/hide filter detail for debugging.
includeCalculationNames	Show/hide filter detail for debugging.
includeRawValue	Show/hide filter detail for debugging.

Value

A HTML widget.

Examples

```
# See the Shiny vignette in this package for examples.
```

---

pivottablerOutput *Standard function for Shiny scaffolding.*

Description

Standard function for Shiny scaffolding.

Usage

```
pivottablerOutput(outputId, width = "100%", height = "100%")
```

Arguments

outputId	The id of the html element that will contain the htmlwidget.
width	The target width of the htmlwidget.
height	The target height of the htmlwidget.

processIdentifier *Handle an identifier that may be illegal (e.g. containing spaces).*

Description

`processIdentifier` is a utility function that wraps an illegal identifier in backticks.

Usage

```
processIdentifier(identifier)
```

Arguments

`identifier`	The identifier that may be illegal.

Value

The identifier wrapped in backticks (if illegal) or unchanged.

processIdentifiers *Handle identifiers that may be illegal (e.g. containing spaces).*

Description

`processIdentifiers` is a utility function that wraps illegal identifiers in backticks.

Usage

```
processIdentifiers(identifiers)
```

Arguments

`identifiers`	The identifiers that may be illegal.

Value

The identifiers wrapped in backticks (if illegal) or unchanged.

pvtperfresults *Performance Comparison Results*

Description

A reference dataset containing the full results of an example performance comparison for different pivot table test cases.

Usage

```
pvtperfresults
```

Format

A data frame with 216 rows and 11 variables:

id A unique identifier for this test case.

evaluationMode The pivot table evaluation mode used for this test case.

rowCount The number of rows in the data frame used for this test case.

cellCount The number of cells in the pivot table used for this test case.

argumentCheckMode The pivot table argument check mode used this test case.

processingLibrary The pivot table processing library used this test case.

description A description of this test case.

completed A logical value indicating whether this test case completed.

userTime The user time for this test case.

systemTime The system time for this test case.

elapsedTime The elapsed time for this test case.

pvtperfsummary *Performance Comparison Summary*

Description

A reference dataset containing summary results of an example performance comparison for different pivot table test cases.

Usage

```
pvtperfsummary
```

Format

A data frame with 36 rows and 18 variables:

id A unique identifier for this test case.

evaluationMode The pivot table evaluation mode used for this test case.

rowCount The number of rows in the data frame used for this test case.

cellCount The number of cells in the pivot table used for this test case.

argumentCheckMode The pivot table argument check mode used this test case.

processingLibrary The pivot table processing library used this test case.

description A description of this test case.

userTimeAvg The average user time for this test case.

systemTimeAvg The average system time for this test case.

elapsedTimeAvg The average elapsed time for this test case.

userTimeMin The minimum user time for this test case.

userTimeMax The maximum user time for this test case.

systemTimeMin The minimum system time for this test case.

systemTimeMax The maximum system time for this test case.

elapsedTimeMin The minimum elapsed time for this test case.

elapsedTimeMax The maximum elapsed time for this test case.

testName A short name for this test case.

testIndex An index for this test case.

---

qhpvt *Quickly render a basic pivot table in HTML.*

Description

The qhpvt function renders a basic pivot table as a HTML widget with one line of R.

Usage

```
qhpvt(dataFrame, rows = NULL, columns = NULL, calculations = NULL,
  theme = NULL, replaceExistingStyles = FALSE, tableStyle = NULL,
  headingStyle = NULL, cellStyle = NULL, totalStyle = NULL, ...)
```

Arguments

dataFrame	The data frame containing the data to be summarised in the pivot table.
rows	A character vector of variable names to be plotted on the rows of the pivot table, or "=" to specify the position of the calculations.
columns	A character vector of variable names to be plotted on the columns of the pivot table, or "=" to specify the position of the calculations.
calculations	One or more summary calculations to use to calculate the values of the cells in the pivot table.
theme	Either the name of a built-in theme (default, largeplain, compact or blank/none) or a list which specifies the default formatting for the table.
replaceExistingStyles	TRUE to completely replace the default styling with the specified tableStyle, headingStyle, cellStyle and/or totalStyle
tableStyle	A list of CSS style declarations that apply to the table.
headingStyle	A list of CSS style declarations that apply to the heading cells in the table.
cellStyle	A list of CSS style declarations that apply to the normal cells in the table.
totalStyle	A list of CSS style declarations that apply to the total cells in the table.
...	Additional arguments, currently format, formats, totals, styleNamePrefix, compatibility and/or argumentCheckMode.

Value

A HTML widget.

Examples

```
qhpvt(bhmtrains, "TOC", "TrainCategory", "n()")
qhpvt(bhmtrains, "TOC", "TrainCategory",
     c("Mean Speed"="mean(SchedSpeedMPH, na.rm=TRUE)",
       "Std Dev Speed"="sd(SchedSpeedMPH, na.rm=TRUE)"),
     formats=list("%.0f", "%.1f"),
     totals=list("TOC"="All TOCs",
       "TrainCategory"="All Categories"))
```

---

qlpvt *Quickly get a Latex representation of a basic pivot table.*

Description

The qlpvt function returns the Latex for a basic pivot table with one line of R.

Usage

```
qlpvt(dataFrame, rows = NULL, columns = NULL, calculations = NULL,
  ...)
```

Arguments

dataFrame	The data frame containing the data to be summarised in the pivot table.
rows	A character vector of variable names to be plotted on the rows of the pivot table, or "=" to specify the position of the calculations.
columns	A character vector of variable names to be plotted on the columns of the pivot table, or "=" to specify the position of the calculations.
calculations	One or more summary calculations to use to calculate the values of the cells in the pivot table.
...	Additional arguments, currently format, formats, totals, argumentCheckMode, compatibility, caption and/or label. See the Latex output vignette for a description of caption and label.

Value

Latex.

Examples

```
qlpvt(bhmtrains, "TOC", "TrainCategory", "n()")
qlpvt(bhmtrains, "TOC", "TrainCategory", "n()",
      caption="my caption", label="mylabel")
```

---

qpvt *Quickly build a basic pivot table.*

Description

The qpvt function builds a basic pivot table with one line of R.

Usage

```
qpvt(dataFrame, rows = NULL, columns = NULL, calculations = NULL,
  theme = NULL, replaceExistingStyles = FALSE, tableStyle = NULL,
  headingStyle = NULL, cellStyle = NULL, totalStyle = NULL, ...)
```

Arguments

dataFrame	The data frame containing the data to be summarised in the pivot table.
rows	A character vector of variable names to be plotted on the rows of the pivot table, or "=" to specify the position of the calculations.
columns	A character vector of variable names to be plotted on the columns of the pivot table, or "=" to specify the position of the calculations.
calculations	One or more summary calculations to use to calculate the values of the cells in the pivot table.

theme	Either the name of a built-in theme (default, largeplain, compact or blank/none) or a list which specifies the default formatting for the table.
replaceExistingStyles	TRUE to completely replace the default styling with the specified tableStyle, headingStyle, cellStyle and/or totalStyle
tableStyle	A list of CSS style declarations that apply to the table.
headingStyle	A list of CSS style declarations that apply to the heading cells in the table.
cellStyle	A list of CSS style declarations that apply to the normal cells in the table.
totalStyle	A list of CSS style declarations that apply to the total cells in the table.
...	Additional arguments, currently format, formats, totals, compatibility and/or argumentCheckMode.

Value

A pivot table.

Examples

```
qpvt(bhmtrains, "TOC", "TrainCategory", "n()")
qpvt(bhmtrains, c("=", "TOC"), c("TrainCategory", "PowerType"),
     c("Number of Trains"="n()",
       "Maximum Speed"="max(SchedSpeedMPH, na.rm=TRUE)"))
```

---

renderBasicTable *Output a table into a package vignette.*

Description

renderBasicTable is utility function that renders a basic table into a package vignette. This function is primarily intended for internal use by the pivottabler package.

Usage

```
renderBasicTable(matrix = NULL, stylePrefix = NULL,
  rowNamesAsHeader = FALSE)
```

Arguments

matrix	Tabular data to render.
stylePrefix	Text prefix for CSS style declarations.
rowNamesAsHeader	Include row names in output.

Value

A basic table rendered as a HTML widget.

Examples

```
renderBasicTable(matrix(c(1:12), nrow=3))
```

---

renderPivottabler *Standard function for Shiny scaffolding.*

Description

Standard function for Shiny scaffolding.

Usage

```
renderPivottabler(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	The R expression to execute and render in the Shiny web application.
env	Standard shiny argument for a render function.
quoted	Standard shiny argument for a render function.

---

skipExportingValue *Should the current value be skipped during export?*

Description

skipExportingValue is a utility function that returns true if the current value should be skipped when exporting.

Usage

```
skipExportingValue(rawValue, exportOptions)
```

Arguments

rawValue	The value to check.
exportOptions	A list of options controlling export behaviour.

Value

TRUE or FALSE indicating whether the current value should be skipped.

`trainstations` *Train Stations*

Description

A reference dataset listing the codes, names and locations of trains stations in Great Britain.

Usage

```
trainstations
```

Format

A data frame with 2568 rows and 7 variables:

CrsCode 3-letter code for the station

StationName The name of the station

OsEasting The UK Ordnance Survey Easting coordinate for the station

OsNorthing The UK Ordnance Survey Northing coordinate for the station

GridReference Grid reference for the station

Latitude Latitude of the station location

Longitude Longitude of the station location

Source

http://www.recenttraintimes.co.uk/

Index