Package ‘databook’

June 13, 2024

Title What the Package Does (One Line, Title Case)

Version 0.0.0.9000

Description What the package does (one paragraph).

License LGPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports data.table,
dplyr,
magrittr,
plyr,
R6,
reshape2,
tibble,
zoo,
purrr,
clipr,
lazyeval,
sjlabelled,
sjmisc,
lubridate

R topics documented:

DataSheet 1

Index **33**

`DataSheet` *DataSheet Class*

Description

DataSheet Class

DataSheet Class

Details

An R6 class to handle and manage a data frame with associated metadata, filters, and various settings.

Methods

`set_data(new_data, messages, check_names)` Sets the data for the DataSheet object.

`set_changes(new_changes)` Sets the changes for the DataSheet object.

`set_filters(new_filters)` Sets the filters for the DataSheet object.

`set_column_selections(new_column_selections)` Sets the column selections for the DataSheet object.

`set_meta(new_meta)` Sets the metadata for the DataSheet object.

`clear_metadata()` Clears the metadata for the DataSheet object.

`clear_variables_metadata()` Clears the variables metadata for the DataSheet object.

`add_defaults_meta()` Adds default metadata to the DataSheet object.

`add_defaults_variables_metadata(column_names)` Adds default variables metadata to the DataSheet object.

`set_objects(new_objects)` Sets the objects for the DataSheet object.

`set_calculations(new_calculations)` Sets the calculations for the DataSheet object.

`set_keys(new_keys)` Sets the keys for the DataSheet object.

`set_comments(new_comments)` Sets the comments for the DataSheet object.

`append_to_metadata(label, value)` Appends to the metadata of the DataSheet object.

`is_metadata(label)` Checks if a label is in the metadata of the DataSheet object.

`set_data_changed(new_val)` Set the data_changed flag.

`set_variables_metadata_changed(new_val)` Set the variables_metadata_changed flag.

`set_metadata_changed(new_val)` Set the metadata_changed flag.

`get_data_frame(convert_to_character, include_hidden_columns, use_current_filter, use_column_selec`
Get the data frame with various options for filtering, column selection, and attribute handling.

`get_variables_metadata(data_type, convert_to_character, property, column, error_if_no_property, di`
Get the metadata for the variables in the data frame.

`get_column_data_types(columns)` Get the data types of the specified columns.

`get_column_labels(columns)` Get the labels of the specified columns.

`get_data_frame_label(use_current_filter)` Get the label of the data frame.

`clear_variables_metadata()` Clear the variables metadata.

`get_metadata(label, include_calculated, excluded_not_for_display)` Get the metadata for the data frame.

`get_changes()` Get the changes made to the data frame.

`get_calculations()` Get the calculations applied to the data frame.

`get_calculation_names(as_list, excluded_items)` Get the names of the calculations applied to the data frame.

`add_columns_to_data(col_name, col_data, use_col_name_as_prefix, hidden, before, adjacent_column, nu`
Add new columns to the data frame.

`get_columns_from_data(col_names, force_as_data_frame, use_current_filter, use_column_selection, re`
Get the data for the specified columns.

`anova_tables(x_col_names, y_col_name, signif.stars, sign_level, means)` Generate ANOVA tables for the specified columns.

`cor(x_col_names, y_col_name, use, method)` Calculate the correlation between specified columns.

`rename_column_in_data(curr_col_name, new_col_name, label, type, .fn, .cols, new_column_names_df, ne` Renames a column in the data.

`remove_columns_in_data(cols, allow_delete_all)` Removes specified columns from the data.

`replace_value_in_data(col_names, rows, old_value, old_is_missing, start_value, end_value, new_value` Replaces values in the specified columns and rows.

`paste_from_clipboard(col_names, start_row_pos, first_clip_row_is_header, clip_board_text)` Pastes data from the clipboard into the specified columns and rows.

`append_to_metadata(property, new_value)` Appends a new value to the metadata of the data.

`append_to_variables_metadata(col_names, property, new_val)` Appends a new value to the variables metadata.

`append_to_changes(value)` Appends a value to the changes list.

`is_metadata(str)` Checks if a string is in the metadata.

`is_variables_metadata(str, col, return_vector)` Checks if a string is in the variables metadata.

`add_defaults_meta()` Adds default values to the metadata.

`add_defaults_variables_metadata(column_names)` Adds default values to the variables metadata for the specified columns.

`remove_rows_in_data(row_names)` Removes the specified rows from the data.

`get_next_default_column_name(prefix)` Gets the next default column name based on the given prefix.

`reorder_columns_in_data(col_order)` Reorders the columns in the data based on the given order.

`insert_row_in_data(start_row, row_data, number_rows, before)` Inserts new rows into the data at the specified position.

`get_data_frame_length(use_current_filter)` Gets the length of the data frame.

`get_factor_data_frame(col_name, include_levels, include_NA_level)` Gets the data frame for a factor column with optional inclusion of levels and NA level.

`get_column_factor_levels(col_name)` Gets the factor levels for the specified column.

`sort_dataframe(col_names, decreasing, na.last, by_row_names, row_names_as_numeric)` Sorts the data frame based on the specified columns.

`convert_column_to_type(col_names, to_type, factor_values, set_digits, set_decimals, keep_attr, ign` Converts the specified columns to the given type.

`copy_columns(col_names)` Copies the specified columns in the data.

`drop_unused_factor_levels(col_name)` Drops unused factor levels in the specified column.

`set_factor_levels(col_name, new_labels, new_levels, set_new_labels)` Sets the factor levels for the specified column.

`edit_factor_level(col_name, old_level, new_level)` Edits a factor level in the specified column.

`set_factor_reference_level(col_name, new_ref_level)` Sets the reference level for a factor column.

`reorder_factor_levels(col_name, new_level_names)` Reorders the factor levels for the specified column.

`get_column_count(use_column_selection)` Gets the count of columns in the data frame.

`get_column_names(as_list, include, exclude, excluded_items, max_no, use_current_column_selection)` Gets the names of the columns in the data frame.

`get_data_type(col_name)` Gets the data type of the specified column.

`set_hidden_columns(col_names)` Sets the specified columns as hidden.

`unhide_all_columns()` Unhides all columns.

`set_row_names(row_names)` Sets the row names of the data.

`set_col_names(col_names)` Sets the column names of the data.

`get_row_names()` Gets the row names of the data.

`get_dim_dataframe()` Gets the dimensions of the data frame.

`set_protected_columns(col_names)` Sets the specified columns as protected.

`add_filter(filter, filter_name, replace, set_as_current, na.rm, is_no_filter, and_or, inner_not, out` Adds a filter to the data.

`add_filter_as_levels(filter_levels, column)` Adds multiple filters based on the levels of a specified column.

`get_current_filter()` Gets the current filter applied to the data.

`set_current_filter(filter_name)` Sets the current filter for the data.

`get_filter_names(as_list, include, exclude, excluded_items)` Gets the names of all filters.

`get_filter(filter_name)` Gets a specific filter by name.

`get_filter_as_logical(filter_name)` Gets the logical vector of a filter.

`get_filter_column_names(filter_name)` Gets the column names used in a filter.

`get_current_filter_column_names()` Gets the column names used in the current filter.

`filter_applied()` Checks if a filter is applied.

`remove_current_filter()`

`filter_string(filter_name)` Returns the string representation of a filter.

`get_filter_as_instat_calculation(filter_name)` Returns the filter as an instat calculation object.

`add_column_selection(column_selection, name, replace, set_as_current, is_everything, and_or)` Adds a column selection to the data.

`get_current_column_selection()` Gets the current column selection applied to the data.

`set_current_column_selection(name)` Sets the current column selection for the data.

`get_column_selection_names(as_list, include, exclude, excluded_items)` Gets the names of all column selections.

`get_column_selection(name)` Gets a specific column selection by name.

`get_column_selection_column_names(name)` Gets the column names used in a column selection.

`get_column_selected_column_names(column_selection_name)` Gets the selected column names for a given column selection name.

`column_selection_applied()` Checks if a column selection is applied.

`remove_current_column_selection()` Removes the current column selection.

`get_variables_metadata_fields(as_list, include, exclude, excluded_items)` Gets the fields of the variables metadata.

`add_object(object_name, object_type_label, object_format, object)` Adds an object with its metadata to the list of objects.

`get_object_names(object_type_label, as_list)` Gets the names of objects of a specified type.

`get_objects(object_type_label)` Gets objects of a specified type.

`get_object(object_name)` Gets a specific object by name.

`rename_object(object_name, new_name, object_type)` Renames an object.

`delete_objects(data_name, object_names, object_type)` Deletes specified objects.

`reorder_objects(new_order)` Reorders the objects.

`data_clone(include_objects, include_metadata, include_logs, include_filters, include_column_select` Clones the data with specified attributes included or excluded.

`freeze_columns(column)` Freezes the specified columns.

`unfreeze_columns()` Unfreezes all columns.

`add_key(col_names, key_name)` Adds a key with specified columns.

`is_key(col_names)` Checks if specified columns form a key.

`has_key()` Checks if there is a key in the data.

`get_keys(key_name)` Gets the keys of the data.

`remove_key(key_name)` Removes a specified key.

`get_comments(comment_id)` Gets the comments for the data.

`remove_comment(key_name)` Removes a comment.

`set_structure_columns(struc_type_1, struc_type_2, struc_type_3)` Sets the structure columns of the data.

`add_dependent_columns(columns, dependent_cols)` Adds dependent columns to the specified columns.

`set_column_colours(columns, colours)` Sets the colours of the specified columns.

`has_colours(columns)` Checks if the specified columns have colours.

`set_column_colours_by_metadata(data_name, columns, property)` Sets the colours of columns based on metadata property.

`remove_column_colours()` Removes the colours of the columns.

`graph_one_variable(columns, numeric, categorical, output, free_scale_axis, ncol, coord_flip, ...)` Creates a graph for a single variable.

`make_date_yearmonthday(year, month, day, f_year, f_month, f_day, year_format, month_format)` Creates a date from year, month, and day columns.

`make_date_yeardoy(year, doy, base, doy_typical_length)` Creates a date from year and day of year columns.

Active bindings

`data_changed` Logical, indicates if the data has changed. If setting a value, `new_value` must be `TRUE` or `FALSE`.

`metadata_changed` Logical, indicates if the metadata has changed. If setting a value, `new_value` must be `TRUE` or `FALSE`.

`variables_metadata_changed` Logical, indicates if the variables metadata has changed. If setting a value, `new_value` must be `TRUE` or `FALSE`.

`current_filter` A list representing the current filter. If setting a value, `filter` must be a list.

`current_column_selection` A list representing the current column selection. If setting a value, `column_selection` must be a list.

Active bindings

`data_changed` Logical, indicates if the data has changed. If setting a value, new_value must be TRUE or FALSE.

`metadata_changed` Logical, indicates if the metadata has changed. If setting a value, new_value must be TRUE or FALSE.

`variables_metadata_changed` Logical, indicates if the variables metadata has changed. If setting a value, new_value must be TRUE or FALSE.

`current_filter` A list representing the current filter. If setting a value, filter must be a list.

`current_column_selection` A list representing the current column selection. If setting a value, column_selection must be a list.

Methods

Public methods:

- `DataSheet$new()`
- `DataSheet$set_data()`
- `DataSheet$set_meta()`
- `DataSheet$clear_metadata()`
- `DataSheet$set_changes()`
- `DataSheet$set_filters()`
- `DataSheet$set_column_selections()`
- `DataSheet$set_objects()`
- `DataSheet$set_calculations()`
- `DataSheet$set_keys()`
- `DataSheet$set_comments()`
- `DataSheet$set_data_changed()`
- `DataSheet$set_variables_metadata_changed()`
- `DataSheet$set_metadata_changed()`
- `DataSheet$get_data_frame()`
- `DataSheet$get_variables_metadata()`
- `DataSheet$get_column_data_types()`
- `DataSheet$get_column_labels()`
- `DataSheet$get_data_frame_label()`
- `DataSheet$clear_variables_metadata()`
- `DataSheet$get_metadata()`
- `DataSheet$get_changes()`
- `DataSheet$get_calculations()`
- `DataSheet$get_calculation_names()`
- `DataSheet$add_columns_to_data()`
- `DataSheet$get_columns_from_data()`
- `DataSheet$anova_tables()`
- `DataSheet$cor()`
- `DataSheet$rename_column_in_data()`
- `DataSheet$remove_columns_in_data()`
- `DataSheet$replace_value_in_data()`
- `DataSheet$paste_from_clipboard()`

- DataSheet$append_to_metadata()
- DataSheet$append_to_variables_metadata()
- DataSheet$append_to_changes()
- DataSheet$is_metadata()
- DataSheet$is_variables_metadata()
- DataSheet$add_defaults_meta()
- DataSheet$add_defaults_variables_metadata()
- DataSheet$remove_rows_in_data()
- DataSheet$get_next_default_column_name()
- DataSheet$reorder_columns_in_data()
- DataSheet$insert_row_in_data()
- DataSheet$get_data_frame_length()
- DataSheet$get_factor_data_frame()
- DataSheet$get_column_factor_levels()
- DataSheet$sort_dataframe()
- DataSheet$convert_column_to_type()
- DataSheet$copy_columns()
- DataSheet$drop_unused_factor_levels()
- DataSheet$set_factor_levels()
- DataSheet$edit_factor_level()
- DataSheet$set_factor_reference_level()
- DataSheet$reorder_factor_levels()
- DataSheet$get_column_count()
- DataSheet$get_column_names()
- DataSheet$get_data_type()
- DataSheet$set_hidden_columns()
- DataSheet$unhide_all_columns()
- DataSheet$set_row_names()
- DataSheet$set_col_names()
- DataSheet$get_row_names()
- DataSheet$get_dim_dataframe()
- DataSheet$set_protected_columns()
- DataSheet$add_filter()
- DataSheet$add_filter_as_levels()
- DataSheet$get_current_filter()
- DataSheet$set_current_filter()
- DataSheet$get_filter_names()
- DataSheet$get_filter()
- DataSheet$get_filter_as_logical()
- DataSheet$get_filter_column_names()
- DataSheet$get_current_filter_column_names()
- DataSheet$filter_applied()
- DataSheet$remove_current_filter()
- DataSheet$filter_string()
- DataSheet$get_filter_as_instat_calculation()

- DataSheet$add_column_selection()
- DataSheet$get_current_column_selection()
- DataSheet$set_current_column_selection()
- DataSheet$get_column_selection_names()
- DataSheet$get_column_selection()
- DataSheet$get_column_selection_column_names()
- DataSheet$get_column_selected_column_names()
- DataSheet$column_selection_applied()
- DataSheet$remove_current_column_selection()
- DataSheet$get_variables_metadata_fields()
- DataSheet$add_object()
- DataSheet$get_object_names()
- DataSheet$get_objects()
- DataSheet$get_object()
- DataSheet$rename_object()
- DataSheet$delete_objects()
- DataSheet$reorder_objects()
- DataSheet$data_clone()
- DataSheet$freeze_columns()
- DataSheet$unfreeze_columns()
- DataSheet$add_key()
- DataSheet$is_key()
- DataSheet$has_key()
- DataSheet$get_keys()
- DataSheet$remove_key()
- DataSheet$get_comments()
- DataSheet$remove_comment()
- DataSheet$set_structure_columns()
- DataSheet$add_dependent_columns()
- DataSheet$set_column_colours()
- DataSheet$has_colours()
- DataSheet$set_column_colours_by_metadata()
- DataSheet$remove_column_colours()
- DataSheet$graph_one_variable()
- DataSheet$make_date_yearmonthday()
- DataSheet$make_date_yeardoy()
- DataSheet$clone()

Method `new()`**:** Create a new DataSheet object.

Usage:

```
DataSheet$new(
  data = data.frame(),
  data_name = "",
  variables_metadata = data.frame(),
  metadata = list(),
  imported_from = "",
  messages = TRUE,
  convert = TRUE,
  create = TRUE,
  start_point = 1,
  filters = list(),
  column_selections = list(),
  objects = list(),
  calculations = list(),
  keys = list(),
  comments = list(),
  keep_attributes = TRUE
)
```

Arguments:

`data` A data frame to be managed by the DataSheet object. Default is an empty data frame.

`data_name` A character string for the name of the data set. Default is an empty string.

`variables_metadata` A data frame containing metadata for the variables. Default is an empty data frame.

`metadata` A list containing additional metadata. Default is an empty list.

`imported_from` A character string indicating the source of the data import. Default is an empty string.

`messages` Logical, if TRUE messages will be shown during the setup. Default is TRUE.

`convert` Logical, if TRUE data will be converted. Default is TRUE.

`create` Logical, if TRUE the data will be created. Default is TRUE.

`start_point` Numeric, the starting point for default naming. Default is 1.

`filters` A list of filters to be applied to the data. Default is an empty list.

`column_selections` A list of column selections. Default is an empty list.

`objects` A list of objects associated with the data. Default is an empty list.

`calculations` A list of calculations to be performed on the data. Default is an empty list.

`keys` A list of keys for the data. Default is an empty list.

`comments` A list of comments associated with the data. Default is an empty list.

`keep_attributes` Logical, if TRUE attributes will be kept. Default is TRUE.

Returns: A new `DataSheet` object.

Method `set_data()`**:** Sets the data for the DataSheet object. Accepts various data types and converts them to a data frame.

Usage:

```
DataSheet$set_data(new_data, messages = TRUE, check_names = TRUE)
```

Arguments:

`new_data` The new data to be set. It can be a matrix, tibble, data.table, ts object, array, or vector.

`messages` Logical, if TRUE, messages will be shown during the data setup. Default is TRUE.

`check_names` Logical, if TRUE, column names will be checked and made valid if necessary. Default is TRUE.

Returns: The DataSheet object with the updated data.

Method `set_meta()`**:** Sets the metadata for the DataSheet object.

Usage:

```
DataSheet$set_meta(new_meta)
```

Arguments:

`new_meta` A list containing the new metadata.

Method `clear_metadata()`**:** Clears the metadata for the DataSheet object.

Usage:

```
DataSheet$clear_metadata()
```

Method `set_changes()`**:** Sets the changes for the DataSheet object.

Usage:

```
DataSheet$set_changes(new_changes)
```

Arguments:

`new_changes` A list containing the new changes.

Method `set_filters()`**:** Sets the filters for the DataSheet object.

Usage:

```
DataSheet$set_filters(new_filters)
```

Arguments:

`new_filters` A list containing the new filters.

Method `set_column_selections()`**:** Sets the column selections for the DataSheet object.

Usage:

```
DataSheet$set_column_selections(new_column_selections)
```

Arguments:

`new_column_selections` A list containing the new column selections.

Method `set_objects()`**:** Sets the objects for the DataSheet object.

Usage:

```
DataSheet$set_objects(new_objects)
```

Arguments:

`new_objects` A list containing the new objects.

Method `set_calculations()`**:** Sets the calculations for the DataSheet object.

Usage:

```
DataSheet$set_calculations(new_calculations)
```

Arguments:

`new_calculations` A list containing the new calculations.

Method `set_keys()`**:** Sets the keys for the DataSheet object.

Usage:

```
DataSheet$set_keys(new_keys)
```

Arguments:

`new_keys` A list containing the new keys.

Method `set_comments()`**:** Sets the comments for the DataSheet object.

Usage:

```
DataSheet$set_comments(new_comments)
```

Arguments:

`new_comments` A list containing the new comments.

Method `set_data_changed():` Set the data_changed flag.

Usage:

```
DataSheet$set_data_changed(new_val)
```

Arguments:

`new_val` Logical, new value for the data_changed flag.

Method `set_variables_metadata_changed():` Set the variables_metadata_changed flag.

Usage:

```
DataSheet$set_variables_metadata_changed(new_val)
```

Arguments:

`new_val` Logical, new value for the variables_metadata_changed flag.

Method `set_metadata_changed():` Set the metadata_changed flag.

Usage:

```
DataSheet$set_metadata_changed(new_val)
```

Arguments:

`new_val` Logical, new value for the metadata_changed flag.

Method `get_data_frame():` Get the data frame with various options for filtering, column selection, and attribute handling.

Usage:

```
DataSheet$get_data_frame(
  convert_to_character = FALSE,
  include_hidden_columns = TRUE,
  use_current_filter = TRUE,
  use_column_selection = TRUE,
  filter_name = "",
  column_selection_name = "",
  stack_data = FALSE,
  remove_attr = FALSE,
  retain_attr = FALSE,
  max_cols,
  max_rows,
  drop_unused_filter_levels = FALSE,
  start_row,
  start_col,
  ...
)
```

Arguments:

`convert_to_character` Logical, if TRUE converts the data to character format.

`include_hidden_columns` Logical, if TRUE includes hidden columns in the output.

`use_current_filter` Logical, if TRUE uses the current filter applied to the data.

`use_column_selection` Logical, if TRUE uses the current column selection.

`filter_name` Character, specifies the name of the filter to use.

`column_selection_name` Character, specifies the name of the column selection to use.

`stack_data` Logical, if TRUE stacks the data.

`remove_attr` Logical, if TRUE removes certain attributes from the data.

`retain_attr` Logical, if TRUE retains certain attributes in the data.

`max_cols` Numeric, specifies the maximum number of columns to include in the output.

`max_rows` Numeric, specifies the maximum number of rows to include in the output.

`drop_unused_filter_levels` Logical, if TRUE drops unused levels from factors in the filtered data.

`start_row` Numeric, specifies the starting row for the output.

`start_col` Numeric, specifies the starting column for the output.

`...` Additional arguments passed to internal functions.

Returns: A data frame with the specified options applied.

Method `get_variables_metadata()`: Get the metadata for the variables in the data frame.

Usage:

```
DataSheet$get_variables_metadata(
  data_type = "all",
  convert_to_character = FALSE,
  property,
  column,
  error_if_no_property = TRUE,
  direct_from_attributes = FALSE,
  use_column_selection = TRUE
)
```

Arguments:

`data_type` Character, the data type to filter by. Default is "all".

`convert_to_character` Logical, if TRUE converts the metadata to character format.

`property` Character, the property of the metadata to retrieve.

`column` Character, the column to retrieve metadata for.

`error_if_no_property` Logical, if TRUE throws an error if the property is not found.

`direct_from_attributes` Logical, if TRUE retrieves metadata directly from attributes.

`use_column_selection` Logical, if TRUE uses the current column selection.

Returns: A data frame or list of metadata for the variables.

Method `get_column_data_types()`: Get the data types of the specified columns.

Usage:

```
DataSheet$get_column_data_types(columns)
```

Arguments:

`columns` Character vector, names of the columns to get data types for.

Returns: A character vector of data types for the specified columns.

Method `get_column_labels()`: Get the labels of the specified columns.

Usage:

```
DataSheet$get_column_labels(columns)
```

Arguments:

`columns` Character vector, names of the columns to get labels for.

Returns: A character vector of labels for the specified columns.

Method `get_data_frame_label()`: Get the label of the data frame.

Usage:

```
DataSheet$get_data_frame_label(use_current_filter = FALSE)
```

Arguments:

`use_current_filter` Logical, if TRUE uses the current filter applied to the data.

Returns: A character string representing the label of the data frame.

Method `clear_variables_metadata()`**:** Clear the variables metadata.

Usage:

```
DataSheet$clear_variables_metadata()
```

Method `get_metadata()`**:** Get the metadata for the data frame.

Usage:

```
DataSheet$get_metadata(
  label,
  include_calculated = TRUE,
  excluded_not_for_display = TRUE
)
```

Arguments:

`label` Character, specifies the metadata label to retrieve.

`include_calculated` Logical, if TRUE includes calculated metadata.

`excluded_not_for_display` Logical, if TRUE excludes metadata not for display.

Returns: A list of metadata for the data frame.

Method `get_changes()`**:** Get the changes made to the data frame.

Usage:

```
DataSheet$get_changes()
```

Returns: A list of changes made to the data frame.

Method `get_calculations()`**:** Get the calculations applied to the data frame.

Usage:

```
DataSheet$get_calculations()
```

Returns: A list of calculations applied to the data frame.

Method `get_calculation_names()`**:** Get the names of the calculations applied to the data frame.

Usage:

```
DataSheet$get_calculation_names(as_list = FALSE, excluded_items = c())
```

Arguments:

`as_list` Logical, if TRUE returns the names as a list.

`excluded_items` Character vector, names of calculations to exclude.

Returns: A character vector or list of calculation names.

Method `add_columns_to_data()`**:** Add new columns to the data frame.

Usage:

```
DataSheet$add_columns_to_data(
  col_name = "",
  col_data,
  use_col_name_as_prefix = FALSE,
  hidden = FALSE,
  before,
  adjacent_column = "",
  num_cols,
  require_correct_length = TRUE,
  keep_existing_position = TRUE
)
```

Arguments:

`col_name` Character, name of the new column.

`col_data` Data, the data for the new column.

`use_col_name_as_prefix` Logical, if TRUE uses the column name as a prefix.

`hidden` Logical, if TRUE the new column will be hidden.

`before` Logical, if TRUE adds the new column before the specified adjacent column.

`adjacent_column` Character, name of the adjacent column.

`num_cols` Numeric, number of columns to add.

`require_correct_length` Logical, if TRUE requires the new column to have the correct length.

`keep_existing_position` Logical, if TRUE keeps the existing position of the new column.

Returns: The updated data frame with the new columns added.

Method `get_columns_from_data()`**:** Get the data for the specified columns.

Usage:

```
DataSheet$get_columns_from_data(
  col_names,
  force_as_data_frame = FALSE,
  use_current_filter = TRUE,
  use_column_selection = TRUE,
  remove_labels = FALSE,
  drop_unused_filter_levels = FALSE
)
```

Arguments:

`col_names` Character vector, names of the columns to retrieve.

`force_as_data_frame` Logical, if TRUE forces the output to be a data frame.

`use_current_filter` Logical, if TRUE uses the current filter applied to the data.

`use_column_selection` Logical, if TRUE uses the current column selection.

`remove_labels` Logical, if TRUE removes labels from the data.

`drop_unused_filter_levels` Logical, if TRUE drops unused levels from factors in the filtered data.

Returns: A data frame or vector of the specified columns.

Method `anova_tables()`**:** Generate ANOVA tables for the specified columns.

Usage:

```
DataSheet$anova_tables(
  x_col_names,
  y_col_name,
  signif.stars = FALSE,
  sign_level = FALSE,
  means = FALSE
)
```

Arguments:

`x_col_names` Character vector, names of the columns to use as independent variables.

`y_col_name` Character, name of the dependent variable column.

`signif.stars` Logical, if TRUE includes significance stars in the output.

`sign_level` Logical, if TRUE includes significance levels in the output.

`means` Logical, if TRUE includes means in the output.

Method `cor()`**:** Calculate the correlation between specified columns.

Usage:

```
DataSheet$cor(
  x_col_names,
  y_col_name,
  use = "everything",
  method = c("pearson", "kendall", "spearman")
)
```

Arguments:

`x_col_names` Character vector, names of the columns to use as independent variables.

`y_col_name` Character, name of the dependent variable column.

`use` Character, specifies the handling of missing values. Default is "everything".

`method` Character vector, specifies the correlation method to be used. One of "pearson", "kendall", or "spearman". Default is c("pearson", "kendall", "spearman").

Returns: A matrix of correlation coefficients between the specified columns.

Method `rename_column_in_data()`**:** Rename a column in the data.

Usage:

```
DataSheet$rename_column_in_data(
  curr_col_name = "",
  new_col_name = "",
  label = "",
  type = "single",
  .fn,
  .cols = everything(),
  new_column_names_df,
  new_labels_df,
  ...
)
```

Arguments:

`curr_col_name` Character, the current name of the column.

`new_col_name` Character, the new name for the column.

`label` Character, the label for the column.

`type` Character, the type of renaming to perform.

`.fn` Function, the function to use for renaming.

`.cols` Character, the columns to rename.

`new_column_names_df` Data frame, the new column names.

`new_labels_df` Data frame, the new labels for the columns.

`...` Additional arguments passed to the function.

Method `remove_columns_in_data()`**:** Remove specified columns from the data.

Usage:

```
DataSheet$remove_columns_in_data(cols = c(), allow_delete_all = FALSE)
```

Arguments:

`cols` Character vector, the names of the columns to remove.

`allow_delete_all` Logical, if TRUE, allows deleting all columns.

Method `replace_value_in_data()`**:** Replace values in the specified columns and rows.

Usage:

```
DataSheet$replace_value_in_data(
  col_names,
  rows,
  old_value,
  old_is_missing = FALSE,
  start_value = NA,
  end_value = NA,
  new_value,
  new_is_missing = FALSE,
  closed_start_value = TRUE,
  closed_end_value = TRUE,
  locf = FALSE,
  from_last = FALSE
)
```

Arguments:

`col_names` Character vector, the names of the columns.

`rows` Character vector, the names of the rows.

`old_value` The old value to be replaced.

`old_is_missing` Logical, if TRUE, treats old_value as missing.

`start_value` Numeric, the starting value for the range to replace.

`end_value` Numeric, the ending value for the range to replace.

`new_value` The new value to replace with.

`new_is_missing` Logical, if TRUE, treats new_value as missing.

`closed_start_value` Logical, if TRUE, includes the start value in the range.

`closed_end_value` Logical, if TRUE, includes the end value in the range.

`locf` Logical, if TRUE, uses the last observation carried forward method.

`from_last` Logical, if TRUE, uses the last observation from the end.

Method `paste_from_clipboard()`**:** Paste data from the clipboard into the specified columns and rows.

Usage:

```
DataSheet$paste_from_clipboard(
  col_names,
  start_row_pos = 1,
  first_clip_row_is_header = FALSE,
  clip_board_text
)
```

Arguments:

`col_names` Character vector, the names of the columns.

`start_row_pos` Numeric, the starting row position.

`first_clip_row_is_header` Logical, if TRUE, treats the first row of the clipboard data as a header.

`clip_board_text` Character, the clipboard text data.

Method `append_to_metadata()`**:** Append a new value to the metadata of the data.

Usage:

```
DataSheet$append_to_metadata(property, new_value = "")
```

Arguments:

`property` Character, the property to append to.

`new_value` The new value to append.

Method `append_to_variables_metadata()`**:** Append a new value to the variables metadata.

Usage:

```
DataSheet$append_to_variables_metadata(col_names, property, new_val = "")
```

Arguments:

`col_names` Character vector, the names of the columns.

`property` Character, the property to append to.

`new_val` The new value to append.

Method `append_to_changes()`**:** Append a value to the changes list.

Usage:

```
DataSheet$append_to_changes(value)
```

Arguments:

`value` The value to append.

Method `is_metadata()`**:** Check if a string is in the metadata.

Usage:

```
DataSheet$is_metadata(str)
```

Arguments:

`str` Character, the string to check.

Returns: Logical, TRUE if the string is in the metadata, FALSE otherwise.

Method `is_variables_metadata()`**:** Check if a string is in the variables metadata.

Usage:

```
DataSheet$is_variables_metadata(str, col, return_vector = FALSE)
```

Arguments:

`str` Character, the string to check.

`col` Character, the column to check in.

`return_vector` Logical, if TRUE, returns the result as a vector.

Returns: Logical, TRUE if the string is in the variables metadata, FALSE otherwise.

Method `add_defaults_meta()`**:** Adds default values to the metadata.

Usage:

```
DataSheet$add_defaults_meta()
```

Method `add_defaults_variables_metadata()`**:** Adds default values to the variables metadata for the specified columns.

Usage:

```
DataSheet$add_defaults_variables_metadata(column_names)
```

Arguments:

`column_names` Character vector, the names of the columns.

Method `remove_rows_in_data()`**:** Removes the specified rows from the data.

Usage:

```
DataSheet$remove_rows_in_data(row_names)
```

Arguments:

`row_names` Character vector, the names of the rows to remove.

Method `get_next_default_column_name()`**:** Gets the next default column name based on the given prefix.

Usage:

```
DataSheet$get_next_default_column_name(prefix)
```

Arguments:

`prefix` Character, the prefix for the new column name.

Returns: Character, the next default column name.

Method `reorder_columns_in_data()`**:** Reorders the columns in the data based on the given order.

Usage:

```
DataSheet$reorder_columns_in_data(col_order)
```

Arguments:

`col_order` Character vector, the new order of the columns.

Method `insert_row_in_data()`**:** Inserts new rows into the data at the specified position.

Usage:

```
DataSheet$insert_row_in_data(
  start_row,
  row_data = c(),
  number_rows = 1,
  before = FALSE
)
```

Arguments:

`start_row` Character, the starting row for the new rows.

`row_data` Data frame, the data for the new rows.

`number_rows` Numeric, the number of new rows to insert.

`before` Logical, if TRUE, inserts the new rows before the specified row.

Method `get_data_frame_length()`**:** Gets the length of the data frame.

Usage:

```
DataSheet$get_data_frame_length(use_current_filter = FALSE)
```

Arguments:

`use_current_filter` Logical, if TRUE, uses the current filter.

Returns: Numeric, the length of the data frame.

Method `get_factor_data_frame()`**:** Gets the data frame for a factor column with optional inclusion of levels and NA level.

Usage:

```
DataSheet$get_factor_data_frame(
  col_name = "",
  include_levels = TRUE,
  include_NA_level = FALSE
)
```

Arguments:

`col_name` Character, the name of the factor column.

`include_levels` Logical, if TRUE, includes the levels of the factor.

`include_NA_level` Logical, if TRUE, includes the NA level.

Returns: Data frame, the data frame for the factor column.

Method `get_column_factor_levels()`**:** Gets the factor levels for the specified column.

Usage:

```
DataSheet$get_column_factor_levels(col_name = "")
```

Arguments:

`col_name` Character, the name of the column.

Returns: Character vector, the factor levels for the column.

Method `sort_dataframe()`**:** Sorts the data frame based on the specified columns.

Usage:

```
DataSheet$sort_dataframe(
  col_names = c(),
  decreasing = FALSE,
  na.last = TRUE,
  by_row_names = FALSE,
  row_names_as_numeric = TRUE
)
```

Arguments:

`col_names` Character vector, the names of the columns to sort by.

`decreasing` Logical, if TRUE, sorts in decreasing order.

`na.last` Logical, if TRUE, places NA values last.

`by_row_names` Logical, if TRUE, sorts by row names.

`row_names_as_numeric` Logical, if TRUE, treats row names as numeric values.

Method `convert_column_to_type()`**:** Converts the specified columns to the given type.

Usage:

```
DataSheet$convert_column_to_type(
  col_names = c(),
  to_type,
  factor_values = NULL,
  set_digits,
  set_decimals = FALSE,
  keep_attr = TRUE,
  ignore_labels = FALSE,
  keep.labels = TRUE
)
```

Arguments:

`col_names` Character vector, the names of the columns.

`to_type` Character, the type to convert to.

`factor_values` Character, the factor values to use for conversion.

`set_digits` Numeric, the number of digits to use for conversion.

`set_decimals` Logical, if TRUE, sets the number of decimals.

`keep_attr` Logical, if TRUE, keeps the attributes of the columns.

`ignore_labels` Logical, if TRUE, ignores labels during conversion.

`keep.labels` Logical, if TRUE, keeps labels during conversion.

Method `copy_columns()`**:** Copies the specified columns in the data.

Usage:

```
DataSheet$copy_columns(col_names = "")
```

Arguments:

`col_names` Character vector, the names of the columns to copy.

Method `drop_unused_factor_levels()`**:** Drops unused factor levels in the specified column.

Usage:

```
DataSheet$drop_unused_factor_levels(col_name)
```

Arguments:

`col_name` Character, the name of the column.

Method `set_factor_levels()`**:** Sets the factor levels for the specified column.

Usage:

```
DataSheet$set_factor_levels(
  col_name,
  new_labels,
  new_levels,
  set_new_labels = TRUE
)
```

Arguments:

`col_name` Character, the name of the column.

`new_labels` Character vector, the new labels for the factor levels.

`new_levels` Character vector, the new levels for the factor.

`set_new_labels` Logical, if TRUE, sets the new labels.

Method `edit_factor_level()`**:** Edits the factor level in the specified column.

Usage:

```
DataSheet$edit_factor_level(col_name, old_level, new_level)
```

Arguments:

`col_name` Character, the name of the column.

`old_level` Character, the old factor level.

`new_level` Character, the new factor level.

Method `set_factor_reference_level()`**:** Sets the reference level for a factor column.

Usage:

```
DataSheet$set_factor_reference_level(col_name, new_ref_level)
```

Arguments:

`col_name` Character, the name of the column.

`new_ref_level` Character, the new reference level.

Method `reorder_factor_levels()`**:** Reorders the factor levels in the specified column.

Usage:

```
DataSheet$reorder_factor_levels(col_name, new_level_names)
```

Arguments:

`col_name` Character, the name of the column.

`new_level_names` Character vector, the new order of the factor levels.

Method `get_column_count()`**:** Gets the number of columns in the data.

Usage:

```
DataSheet$get_column_count(use_column_selection = FALSE)
```

Arguments:

`use_column_selection` Logical, if TRUE, uses the current column selection.

Returns: Numeric, the number of columns in the data.

Method `get_column_names()`**:** Gets the names of the columns in the data.

Usage:

```
DataSheet$get_column_names(
  as_list = FALSE,
  include = list(),
  exclude = list(),
  excluded_items = c(),
  max_no,
  use_current_column_selection = TRUE
)
```

Arguments:

`as_list` Logical, if TRUE, returns the names as a list.

`include` List, the properties to include.

`exclude` List, the properties to exclude.

`excluded_items` Character vector, the items to exclude.

`max_no` Numeric, the maximum number of columns to return.

`use_current_column_selection` Logical, if TRUE, uses the current column selection.

Returns: Character vector or list, the names of the columns in the data.

Method `get_data_type()`**:** Gets the data type of the specified column.

Usage:

```
DataSheet$get_data_type(col_name = "")
```

Arguments:

`col_name` Character, the name of the column.

Returns: Character, the data type of the column.

Method `set_hidden_columns()`**:** Set the hidden columns in the data.

Usage:

```
DataSheet$set_hidden_columns(col_names = c())
```

Arguments:

`col_names` Character vector, the names of the columns to hide.

Method `unhide_all_columns()`**:** Unhide all columns in the data.

Usage:

```
DataSheet$unhide_all_columns()
```

Method `set_row_names()`**:** Set the row names of the data frame.

Usage:

```
DataSheet$set_row_names(row_names)
```

Arguments:

`row_names` Character vector, the new row names.

Method `set_col_names()`**:** Set the column names of the data frame.

Usage:

```
DataSheet$set_col_names(col_names)
```

Arguments:

`col_names` Character vector, the new column names.

Method `get_row_names()`**:** Get the row names of the data frame.

Usage:

```
DataSheet$get_row_names()
```

Returns: Character vector, the row names of the data frame.

Method `get_dim_dataframe()`**:** Get the dimensions of the data frame.

Usage:

```
DataSheet$get_dim_dataframe()
```

Returns: Numeric vector, the dimensions of the data frame.

Method `set_protected_columns()`**:** Set the protected columns in the data.

Usage:

```
DataSheet$set_protected_columns(col_names)
```

Arguments:

`col_names` Character vector, the names of the columns to protect.

Method `add_filter()`**:** Add a filter to the data.

Usage:

```
DataSheet$add_filter(
  filter,
  filter_name = "",
  replace = TRUE,
  set_as_current = FALSE,
  na.rm = TRUE,
  is_no_filter = FALSE,
  and_or = "&",
  inner_not = FALSE,
  outer_not = FALSE
)
```

Arguments:

`filter` List, the filter conditions.

`filter_name` Character, the name of the filter.

`replace` Logical, if TRUE, replaces an existing filter with the same name.

`set_as_current` Logical, if TRUE, sets the filter as the current filter.

`na.rm` Logical, if TRUE, removes NA values.

`is_no_filter` Logical, if TRUE, specifies that no filter is applied.

`and_or` Character, specifies the logical operator for combining conditions.

`inner_not` Logical, if TRUE, applies negation to the inner condition.

`outer_not` Logical, if TRUE, applies negation to the outer condition.

Method `add_filter_as_levels()`**:** Add filters based on levels of a column.

Usage:

```
DataSheet$add_filter_as_levels(filter_levels, column)
```

Arguments:

`filter_levels` Character vector, the levels to create filters for.

`column` Character, the name of the column.

Method `get_current_filter()`**:** Get the current filter.

Usage:

```
DataSheet$get_current_filter()
```

Returns: List, the current filter.

Method `set_current_filter()`**:** Set the current filter by name.

Usage:

```
DataSheet$set_current_filter(filter_name = "")
```

Arguments:

`filter_name` Character, the name of the filter to set as current.

Method `get_filter_names()`**:** Get the names of all filters.

Usage:

```
DataSheet$get_filter_names(
  as_list = FALSE,
  include = list(),
  exclude = list(),
  excluded_items = c()
)
```

Arguments:

`as_list` Logical, if TRUE, returns the names as a list.

`include` List, the properties to include.

`exclude` List, the properties to exclude.

`excluded_items` Character vector, the items to exclude.

Returns: Character vector or list, the names of the filters.

Method `get_filter()`**:** Get a specific filter by name.

Usage:

```
DataSheet$get_filter(filter_name)
```

Arguments:

`filter_name` Character, the name of the filter.

Returns: List, the specified filter.

Method `get_filter_as_logical()`**:** Get the filter as a logical vector.

Usage:

```
DataSheet$get_filter_as_logical(filter_name)
```

Arguments:

`filter_name` Character, the name of the filter.

Returns: Logical vector, the filter applied as a logical vector.

Method `get_filter_column_names()`**:** Get the column names used in a specific filter.

Usage:

```
DataSheet$get_filter_column_names(filter_name)
```

Arguments:

`filter_name` Character, the name of the filter.

Returns: Character vector, the column names used in the filter.

Method `get_current_filter_column_names()`**:** Get the column names used in the current filter.

Usage:

```
DataSheet$get_current_filter_column_names()
```

Returns: Character vector, the column names used in the current filter.

Method `filter_applied()`**:** Check if a filter is applied.

Usage:

```
DataSheet$filter_applied()
```

Returns: Logical, TRUE if a filter is applied, FALSE otherwise.

Method `remove_current_filter()`**:** Remove the current filter.

Usage:

```
DataSheet$remove_current_filter()
```

Method `filter_string()`**:** Get the filter as a string.

Usage:

```
DataSheet$filter_string(filter_name)
```

Arguments:

`filter_name` Character, the name of the filter.

Returns: Character, the filter as a string.

Method `get_filter_as_instat_calculation()`**:** Get the filter as an instat calculation.

Usage:

```
DataSheet$get_filter_as_instat_calculation(filter_name)
```

Arguments:

`filter_name` Character, the name of the filter.

Returns: Instat calculation, the filter as an instat calculation.

Method `add_column_selection()`**:** Add a column selection to the data.

Usage:

```
DataSheet$add_column_selection(
  column_selection,
  name = "",
  replace = TRUE,
  set_as_current = FALSE,
  is_everything = FALSE,
  and_or = "|"
)
```

Arguments:

`column_selection` List, the column selection conditions.

`name` Character, the name of the column selection.

`replace` Logical, if TRUE, replaces an existing column selection with the same name.

`set_as_current` Logical, if TRUE, sets the column selection as the current selection.

`is_everything` Logical, if TRUE, selects all columns.

`and_or` Character, specifies the logical operator for combining conditions.

Method `get_current_column_selection()`**:** Get the current column selection.

Usage:

```
DataSheet$get_current_column_selection()
```

Returns: List, the current column selection.

Method `set_current_column_selection()`**:** Set the current column selection by name.

Usage:

```
DataSheet$set_current_column_selection(name = "")
```

Arguments:

`name` Character, the name of the column selection to set as current.

Method `get_column_selection_names()`**:** Get the names of all column selections.

Usage:

```
DataSheet$get_column_selection_names(
  as_list = FALSE,
  include = list(),
  exclude = list(),
  excluded_items = c()
)
```

Arguments:

`as_list` Logical, if TRUE, returns the names as a list.

`include` List, the properties to include.

`exclude` List, the properties to exclude.

`excluded_items` Character vector, the items to exclude.

Returns: Character vector or list, the names of the column selections.

Method `get_column_selection()`**:** Get a specific column selection by name.

Usage:

```
DataSheet$get_column_selection(name)
```

Arguments:

`name` Character, the name of the column selection.

Returns: List, the specified column selection.

Method `get_column_selection_column_names()`**:** Get the column names selected by a specific column selection.

Usage:

```
DataSheet$get_column_selection_column_names(name)
```

Arguments:

`name` Character, the name of the column selection.

Returns: Character vector, the column names selected by the column selection.

Method `get_column_selected_column_names()`**:** Get the column names selected by the current column selection.

Usage:

```
DataSheet$get_column_selected_column_names(column_selection_name = "")
```

Arguments:

`column_selection_name` Character, the name of the column selection.

Returns: Character vector, the column names selected by the current column selection.

Method `column_selection_applied()`**:** Check if a column selection is applied.

Usage:

```
DataSheet$column_selection_applied()
```

Returns: Logical, TRUE if a column selection is applied, FALSE otherwise.

Method `remove_current_column_selection()`**:** Remove the current column selection.

Usage:

```
DataSheet$remove_current_column_selection()
```

Method `get_variables_metadata_fields()`**:** Get the fields of the variables metadata.

Usage:

```
DataSheet$get_variables_metadata_fields(
  as_list = FALSE,
  include = c(),
  exclude = c(),
  excluded_items = c()
)
```

Arguments:

`as_list` Logical, if TRUE, returns the fields as a list.

`include` Character vector, the fields to include.

`exclude` Character vector, the fields to exclude.

`excluded_items` Character vector, the items to exclude.

Returns: Character vector or list, the fields of the variables metadata.

Method `add_object()`**:** Add an object to the data.

Usage:

```
DataSheet$add_object(object_name, object_type_label, object_format, object)
```

Arguments:

`object_name` Character, the name of the object.

`object_type_label` Character, the type label of the object.

`object_format` Character, the format of the object.

`object` Any, the object to add.

Method `get_object_names()`**:** Get the names of objects.

Usage:

```
DataSheet$get_object_names(object_type_label = NULL, as_list = FALSE)
```

Arguments:

`object_type_label` Character, the type label of the objects to get names for.

`as_list` Logical, if TRUE, returns the names as a list.

Returns: Character vector or list, the names of the objects.

Method `get_objects()`**:** Get objects by type label.

Usage:

```
DataSheet$get_objects(object_type_label = NULL)
```

Arguments:

`object_type_label` Character, the type label of the objects to get.

Returns: List, the objects with the specified type label.

Method `get_object()`**:** Get a specific object by name.

Usage:

```
DataSheet$get_object(object_name)
```

Arguments:

`object_name` Character, the name of the object.

Returns: Any, the specified object.

Method `rename_object()`**:** Rename an object.

Usage:

```
DataSheet$rename_object(object_name, new_name, object_type = "object")
```

Arguments:

`object_name` Character, the current name of the object.

`new_name` Character, the new name for the object.

`object_type` Character, the type of the object.

Method `delete_objects()`**:** Delete objects.

Usage:

```
DataSheet$delete_objects(data_name, object_names, object_type = "object")
```

Arguments:

`data_name` Character, the name of the data.

`object_names` Character vector, the names of the objects to delete.

`object_type` Character, the type of the objects to delete.

Method `reorder_objects()`**:** Reorder objects.

Usage:

```
DataSheet$reorder_objects(new_order)
```

Arguments:

`new_order` Character vector, the new order of the objects.

Method `data_clone()`**:** Clone the data sheet.

Usage:

```
DataSheet$data_clone(
  include_objects = TRUE,
  include_metadata = TRUE,
  include_logs = TRUE,
  include_filters = TRUE,
  include_column_selections = TRUE,
  include_calculations = TRUE,
  include_comments = TRUE,
  ...
)
```

Arguments:

`include_objects` Logical, if TRUE, includes objects in the clone.

`include_metadata` Logical, if TRUE, includes metadata in the clone.

`include_logs` Logical, if TRUE, includes logs in the clone.

`include_filters` Logical, if TRUE, includes filters in the clone.

`include_column_selections` Logical, if TRUE, includes column selections in the clone.

`include_calculations` Logical, if TRUE, includes calculations in the clone.

`include_comments` Logical, if TRUE, includes comments in the clone.

`...` Additional arguments.

Returns: DataSheet, the cloned data sheet.

Method `freeze_columns()`**:** Freeze columns in the data.

Usage:

```
DataSheet$freeze_columns(column)
```

Arguments:

`column` Character, the name of the column to freeze.

Method `unfreeze_columns()`**:** Unfreeze all columns in the data.

Usage:

```
DataSheet$unfreeze_columns()
```

Method `add_key()`**:** Add a key to the data.

Usage:

```
DataSheet$add_key(col_names, key_name)
```

Arguments:

`col_names` Character vector, the names of the columns to use as the key.

`key_name` Character, the name of the key.

Method `is_key()`**:** Check if columns are a key.

Usage:

```
DataSheet$is_key(col_names)
```

Arguments:

`col_names` Character vector, the names of the columns to check.

Returns: Logical, TRUE if the columns are a key, FALSE otherwise.

Method `has_key()`**:** Check if the data has a key.

Usage:

```
DataSheet$has_key()
```

Returns: Logical, TRUE if the data has a key, FALSE otherwise.

Method `get_keys()`**:** Get the keys in the data.

Usage:

```
DataSheet$get_keys(key_name)
```

Arguments:

`key_name` Character, the name of the key to get.

Returns: List, the keys in the data.

Method `remove_key()`**:** Remove a key from the data.

Usage:

```
DataSheet$remove_key(key_name)
```

Arguments:

`key_name` Character, the name of the key to remove.

Method `get_comments()`**:** Get comments in the data.

Usage:

```
DataSheet$get_comments(comment_id)
```

Arguments:

`comment_id` Character, the ID of the comment to get.

Returns: List, the comments in the data.

Method `remove_comment()`**:** Remove a comment from the data.

Usage:

```
DataSheet$remove_comment(key_name)
```

Arguments:

`key_name` Character, the name of the key to remove the comment from.

Method `set_structure_columns()`: Set the structure columns in the data.

Usage:

```
DataSheet$set_structure_columns(struc_type_1, struc_type_2, struc_type_3)
```

Arguments:

`struc_type_1` Character vector, the names of the columns for structure type 1.

`struc_type_2` Character vector, the names of the columns for structure type 2.

`struc_type_3` Character vector, the names of the columns for structure type 3.

Method `add_dependent_columns()`: Add dependent columns to the data.

Usage:

```
DataSheet$add_dependent_columns(columns, dependent_cols)
```

Arguments:

`columns` Character vector, the names of the columns.

`dependent_cols` List, the dependent columns.

Method `set_column_colours()`: Set the colors of the columns in the data.

Usage:

```
DataSheet$set_column_colours(columns, colours)
```

Arguments:

`columns` Character vector, the names of the columns.

`colours` Character vector, the colors to set.

Method `has_colours()`: Check if columns have colors.

Usage:

```
DataSheet$has_colours(columns)
```

Arguments:

`columns` Character vector, the names of the columns.

Returns: Logical, TRUE if the columns have colors, FALSE otherwise.

Method `set_column_colours_by_metadata()`: Set the colors of the columns based on metadata.

Usage:

```
DataSheet$set_column_colours_by_metadata(data_name, columns, property)
```

Arguments:

`data_name` Character, the name of the data.

`columns` Character vector, the names of the columns.

`property` Character, the property to base the colors on.

Method `remove_column_colours()`: Remove the colors from all columns.

Usage:

```
DataSheet$remove_column_colours()
```

Method `graph_one_variable()`: Create a graph for one variable.

Usage:

```
DataSheet$graph_one_variable(
  columns,
  numeric = "geom_boxplot",
  categorical = "geom_bar",
  output = "facets",
  free_scale_axis = FALSE,
  ncol = NULL,
  coord_flip = FALSE,
  ...
)
```

Arguments:

`columns` Character vector, the names of the columns.

`numeric` Character, the geom for numeric columns.

`categorical` Character, the geom for categorical columns.

`output` Character, the output type ("facets", "combine", "single").

`free_scale_axis` Logical, if TRUE, uses a free scale for the axis.

`ncol` Numeric, the number of columns for facets.

`coord_flip` Logical, if TRUE, flips the coordinates.

`...` Additional arguments for the geom functions.

Returns: ggplot2 object, the graph.

Method `make_date_yearmonthday()`**:** Create a date from year, month, and day columns.

Usage:

```
DataSheet$make_date_yearmonthday(
  year,
  month,
  day,
  f_year,
  f_month,
  f_day,
  year_format = "%Y",
  month_format = "%m"
)
```

Arguments:

`year` Character, the name of the year column.

`month` Character, the name of the month column.

`day` Character, the name of the day column.

`f_year` Numeric vector, the year values.

`f_month` Numeric vector, the month values.

`f_day` Numeric vector, the day values.

`year_format` Character, the format of the year.

`month_format` Character, the format of the month.

Returns: Date, the created date.

Method `make_date_yeardoy()`**:** Create a date from year and day-of-year columns.

Usage:

```
DataSheet$make_date_yeardoy(year, doy, base, doy_typical_length = "366")
```

Arguments:

`year` Character, the name of the year column.

`doy` Character, the name of the day-of-year column.

`base` Numeric, the base year.

`doy_typical_length` Character, the typical length of the day-of-year ("365" or "366").

Returns: Date, the created date.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataSheet$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index

DataSheet, 1