Package 'shinyHugePlot'

March 17, 2023

Type Package

Title Efficient Plotting of Large-Sized Data

Version 0.2.3

Maintainer Junta Tagusari <j.tagusari@eng.hokudai.ac.jp>

Language en-US

Depends R (>= 4.2.0), plotly (>= 4.10.0), shiny (>= 1.7.1)

Imports R6 (>= 2.5.1), dplyr (>= 1.0.9), tibble (>= 3.1.7), tidyr (>= 1.2.0), tidyselect (>= 1.1.2), data.table (>= 1.14.2), stringr (>= 1.4.0), nanotime (>= 0.3.6), assertthat (>= 0.2.1), bit64 (>= 4.0.5), purrr (>= 0.3.4), jsonlite (>= 1.8.0), lazyeval (>= 0.2.2), shinyjs (>= 2.1.0), htmltools (>= 0.5.2), rlang (>= 1.0.5)

Suggests testthat

Description A tool to plot data with a large sample size using 'shiny' and 'plotly'. Relatively small samples are obtained from the original data using a specific algorithm. The samples are updated according to a user-defined x range. Jonas Van Der Donckt, Jeroen Van Der Donckt, Emiel Deprost (2022) <https://github.com/predict-idlab/plotly-resampler>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Junta Tagusari [aut, cre, cph], Jonas Van Der Donckt [cph], Jeroen Van Der Donckt [cph], Emiel Deprost [cph]

Repository CRAN

Date/Publication 2023-03-17 12:20:02 UTC

R topics documented:

`aggregator` *R6 base class for the aggregation*

Description

A base class for the aggregation, which defines the structure of the class and is not available on a stand-alone basis.

Format

An `R6::R6Class` object

Active bindings

`parameters` Parameters for the aggregation, returned as a named list. Generate a matrix using x and n_out Apply function for nanotime

Methods

Public methods:

- `aggregator$new()`
- `aggregator$aggregate()`
- `aggregator$set_parameters()`
- `aggregator$clone()`

Method `new()`**:** Constructor of `aggregator`

Usage:

```
aggregator$new(
  ...,
  interleave_gaps = FALSE,
  NA_position = "begin",
  coef_gap = 3,
  accepted_datatype = NULL
)
```

Arguments:

`...` Not used.

`interleave_gaps`, `NA_position`, `coef_gap`, `accepted_datatype` Arguments passed to `self$set_parameters`, optional.

Method `aggregate()`**:** Aggregates the given input and returns samples.

Usage:

```
aggregator$aggregate(x, y, n_out)
```

Arguments:

`x`, `y` Indexes and values that has to be aggregated.

`n_out` Integer or numeric. The number of samples that the aggregated data contains.

Method `set_parameters()`**:** Setting of the parameters for the aggregation

Usage:

```
aggregator$set_parameters(
  ...,
  interleave_gaps,
  NA_position,
  coef_gap,
  accepted_datatype
)
```

Arguments:

`...` Not used.

`interleave_gaps` Boolean, optional. Whether `NA` values should be added when there are gaps / irregularly sampled data. Irregular gaps between samples are determined whether the gap is larger than the median of the sample gaps times the coefficient for detecting irregular gaps. By default, `FALSE`.

`NA_position` Character, optional. Indicates where `NA`s are placed when gaps are detected. If `"end"`, the first point after a gap will be replaced. If `"begin"`, the last point before a gap will be replaced. If `"both"`, both the encompassing gap data points are replaced. This parameter is only effective when `interleave_gaps == TRUE`. By default, `"begin"`.

`coef_gap` Numeric, optional. The coefficient to detect irregular gaps. By default, 3.0.

`accepted_datatype` Character, optional. This parameter indicates the supported data classes. If all data classes are accepted, set it to `NULL`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`custom_func_aggregator` *Aggregation using a user-defined function.*

Description

Arbitrary function can be applied using this aggregation class.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::aggregator` -> `custom_func_aggregator`

Methods

Public methods:

- `custom_func_aggregator$new()`
- `custom_func_aggregator$set_aggregation_func()`
- `custom_func_aggregator$clone()`

Method `new()`**:** Constructor of the Aggregator.

Usage:

```
custom_func_aggregator$new(
  ...,
  aggregation_func,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

`aggregation_func` Function. User-defined function to aggregate data, of which arguments are `x`, `y` and `n_out`.

`interleave_gaps`, `coef_gap`, `NA_position`, `accepted_datatype`, `...` Arguments pass to the constructor of `aggregator` object.

Method `set_aggregation_func()`**:** Set a function to aggregate the data

Usage:

```
custom_func_aggregator$set_aggregation_func(aggregation_func)
```

Arguments:

`aggregation_func` Function. User-defined function to aggregate data, of which arguments are `x`, `y` and `n_out`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
custom_agg_func <- function(x, y, n_out) {
  bin_width <- floor(length(x)/n_out)
  x_idx <- seq(floor(bin_width / 2), bin_width * n_out, bin_width)
  y_mat <- y[1:(bin_width * n_out)] %>%
    matrix(nrow = bin_width)
  y_agg <- apply(y_mat, 2, quantile, probs = 0.25)
  return(list(x = x[x_idx], y = y_agg))
}
data(noise_fluct)
agg <- custom_func_aggregator$new(
  aggregation_func = custom_agg_func, interleave_gaps = TRUE
  )
d_agg <- agg$aggregate(
  x = noise_fluct$time, y = noise_fluct$f500, n_out = 1000
  )
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

---

custom_stat_aggregator

Aggregation which returns arbitrary statistics

Description

This aggregator divides the data into no-overlapping intervals and calculate specific statistical values such as the mean.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::aggregator` -> `custom_stat_aggregator`

Methods

Public methods:

- custom_stat_aggregator$new()
- custom_stat_aggregator$clone()

Method `new()`**:** Constructor of the Aggregator.

Constructor of the Aggregator.

Usage:

```
custom_stat_aggregator$new(
  ...,
  y_func = mean,
  x_mean = TRUE,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

`y_func` Function. Statistical values are calculated using this function. By default, `mean`.

`x_mean` Boolean. Whether using the mean values or not for the x values. If not, the x values that give the specific y values are used. E.g., if you use `max` as the `aggregation_func` and set this argument to `FALSE`, x values that give the maximum y values are used. By default, `TRUE`.

`interleave_gaps`, `coef_gap`, `NA_position`, `accepted_datatype`, `...` Arguments pass to the constructor of `aggregator` object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- custom_stat_aggregator$new(y_func = mean, interleave_gaps = TRUE)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500, 1000)
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

downsampler *R6 class for down-sampling data*

Description

A class for down-sampling data with a large number of samples. An instance contains (the reference of) original data, layout of the figure, and options for aggregating the original data. An interactive plot for displaying large-sized data can be obtained using the figure, down-sampler and its options included in the instance, while making the plot using `shiny_hugeplot` function is easier (see examples). See the super class (`plotly_datahandler`) to find more members to handle the data in `plotly`.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::plotly_datahandler` -> `downsampler`

Active bindings

`downsample_options` Options for aggregating (down-sampling) data registered in this instance.

`n_out_default` Default sample size.

`aggregator_default` Default aggregator instance.

Methods

Public methods:

- `downsampler$new()`
- `downsampler$add_trace()`
- `downsampler$update_trace()`
- `downsampler$set_downsample_options()`
- `downsampler$clone()`

Method `new()`**:** To construct an instance, original data, layout of the figure, and options for aggregating the original data are necessary. The original data and the layout of the figure can be given by providing a `plotly` object (`figure` argument). The options for aggregating the original data can be given by providing an aggregator (`aggregator` argument) and the number of samples (`n_out` argument). See the constructor of the `plotly_datahandler` class for more information on other arguments.

Usage:

```
downsampler$new(
  figure = NULL,
  n_out = 1000L,
  aggregator = min_max_aggregator$new(),
  tz = Sys.timezone(),
  use_light_build = TRUE,
 legend_options = list(name_prefix = "<b style=\"color:sandybrown\">[S]</b> ",
  name_suffix = "", xdiff_prefix = "<i style=\"color:#fc9944\"> ~", xdiff_suffix =
   "</i>")
)
```

Arguments:

`figure`, `legend_options`, `tz`, `use_light_build` Arguments passed to `plotly_datahandler$new`.

`n_out` Integer or numeric. The number of samples shown after down-sampling. By default 1000.

`aggregator` An instance of an R6 class for aggregation. Select an aggregation function. The list of the functions are obtained using `list_aggregators`. By default, `min_max_aggregator$new()`.

Method `add_trace()`**:** Add a new series to the data registered in the instance. If a data frame (`traces_df` argument) compliant with `self$orig_data` is given, it will be added to `self$orig_data`. If attributes to construct a `plotly` object (`...` argument) are given, a data frame is constructed and added. Options for aggregating data can be set using `aggregator` and `n_out` arguments. It is a wrapper of `self$set_trace_data` and `self$set_downsample_options`. See these methods for more information. Note that the traces of the figure are not updated with this method and `self$update_trace` is necessary.

Usage:

```
downsampler$add_trace(..., traces_df = NULL, n_out = NULL, aggregator = NULL)
```

Arguments:

`...`, `traces_df` Arguments passed to `self$set_trace_data` (see the super class of `plotly_datahandler`)

`n_out`, `aggregator` Arguments passed to `self$set_downsample_options`.

Method `update_trace()`**:** Update traces of the figure registered in the instance (`self$figure$x$data`) according to re-layout order (`relayout_order` argument). Using `reset` and `reload` arguments, traces are updated without re-layout orders. It just registers the new traces and returns nothing by default. It returns the new traces if `send_trace` is `TRUE`.

Usage:

```
downsampler$update_trace(
  relayout_order = list(NULL),
  reset = FALSE,
  reload = FALSE,
  send_trace = FALSE
)
```

Arguments:

`relayout_order` Named list. A list generated by `plotlyjs_relayout`, which is obtained using `plotly::event_data`. e.g., If you would like set the range of the 2nd x axis to [10.0, 21.5], `list(`xaxis2.range[0]` = 10.0, `xaxis2.range[1]` = 21.5)`. If you would like reset the range of the 1st x axis, `list(xaxis.autorange = TRUE, xaxis.showspike = TRUE)`.

`reset` Boolean. If it is `TRUE`, all other arguments are neglected and the figure will be reset (all the ranges of x axes are initialized). By default, `FALSE`.

`reload` Boolean. If it is `TRUE`, the ranges of the figure are preserved but the aggregation will be conducted with the current settings. By default, `FALSE`.

`send_trace` Boolean. If it is `TRUE`, a named list will be returned, which contains the indexes of the traces that will be updated (`trace_idx_update`) and the updated traces (`new_trace`). By default, `FALSE`.

Method `set_downsample_options()`**:** In the instance, options for aggregating data are registered as data frame. (see `self$downsample_options`.) Using this method, the options can be set.

Usage:

```
downsampler$set_downsample_options(uid = NULL, n_out = NULL, aggregator = NULL)
```

Arguments:

`uid` Character, optional. The unique id of the trace. If `NULL`, all the options registered in this instance are updated. By default, `NULL`.

`n_out` Numeric or integer, optional. The number of samples output by the aggregator. If `NULL`, the default value registered in this instance is used. By default, `NULL`.

`aggregator` `aggregator` object, optional. An instance that aggregate the data. If `NULL`, the default value registered in this instance is used.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)

# example 1 : Easy method using shiny_hugeplot
shiny_hugeplot(noise_fluct$time, noise_fluct$f500)

# example 2 : Manual method using a downsampler object
fig <- plot_ly(
  x = noise_fluct$time,
  y = noise_fluct$f500,
  type = "scatter",
  mode = "lines"
  ) %>%
  layout(xaxis = list(type = "date")) %>%
  shinyHugePlot::plotly_build_light()

ds <- downsampler$new(
  figure = fig,
  aggregator = min_max_aggregator$new(interleave_gaps = TRUE)
```

```
)

ui <- fluidPage(
  plotlyOutput(outputId = "hp", width = "800px", height = "600px")
)

server <- function(input, output, session) {

  output$hp <- renderPlotly(ds$figure)

  observeEvent(plotly::event_data("plotly_relayout"),{
    updatePlotlyH(session, "hp", plotly::event_data("plotly_relayout"), ds)
  })

}

shinyApp(ui = ui, server = server)


# example 3 : Add another series of which aggregator is different

noise_events <- tibble(
  time = c("2022-11-09 12:25:50", "2022-11-09 12:26:14"),
  level = c(60, 60)
)

ds$add_trace(
  x = noise_events$time, y = noise_events$level, name = "event",
  type = "scatter", mode = "markers",
  aggregator = null_aggregator$new()
)
ds$update_trace(reset = TRUE)

server <- function(input, output, session) {

  output$hp <- renderPlotly(ds$figure)

  observeEvent(plotly::event_data("plotly_relayout"),{
    updatePlotlyH(session, "hp", plotly::event_data("plotly_relayout"), ds)
  })

}

shinyApp(ui = ui, server = server)
```

`eLTTB_aggregator` *Aggregation using local minimum and maximum values, and Largest Triangle Three Buckets (LTTB) method.*

Description

Efficient version off LTTB by first reducing really large data with the `min_max_ovlp_aggregator` and then further aggregating the reduced result with `LTTB_aggregator`.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::aggregator` -> `eLTTB_aggregator`

Public fields

`LTTB` An R6 LTTB_aggregator instance

`minmax` An R6 `min_max_ovlp_aggregator` instance

Methods

Public methods:

- `eLTTB_aggregator$new()`
- `eLTTB_aggregator$clone()`

Method `new()`**:** Constructor of the aggregator.

Usage:

```
eLTTB_aggregator$new(
  ...,
  interleave_gaps,
  coef_gap,
  NA_position,
 accepted_datatype = c("numeric", "integer", "character", "factor", "logical")
)
```

Arguments:

`...` Arguments pass to the constructor of `aggregator`, `LTTB_aggregator` and `min_max_oblp_aggregator` objects.

`interleave_gaps, coef_gap, NA_position, accepted_datatype` Arguments pass to the constructor of `aggregator` object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- eLTTB_aggregator$new(interleave_gaps = TRUE)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500, 1000)
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

---

list_aggregators *Show the aggregation functions*

Description

It displays all the aggregators registered in the package. No arguments are necessary.

Usage

```
list_aggregators()
```

Examples

```
list_aggregators()
```

LTTB_aggregator *Aggregation using Largest Triangle Three Buckets (LTTB) method.*

Description

The LTTB method aggregates the huge samples using the areas of the triangles formed by the samples. Numerical distances are employed in this class, which requires the ratio between x and y values. When the x is datetime, nanosecond is a unit. When the x is factor or character, it will be encoded into numeric codes.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::aggregator` -> `LTTB_aggregator`

Methods

Public methods:

- LTTB_aggregator$new()
- LTTB_aggregator$clone()

Method `new()`**:** Constructor of the aggregator.

Usage:

```
LTTB_aggregator$new(
  ...,
  nt_y_ratio = 1e+09,
  x_y_ratio = 1,
  interleave_gaps,
  coef_gap,
  NA_position,
 accepted_datatype = c("numeric", "integer", "character", "factor", "logical")
)
```

Arguments:

`x_y_ratio`, `nt_y_ratio` Numeric. These parameters set the unit length of the numeric `x` and `nanotime` `x`. For example, setting `x_y_ratio` to 2 is equivalent to assuming 2 is the unit length of `x` (and 1 is always the unit length of `y`). The unit length is employed to calculate the area of the triangles.

`interleave_gaps`, `coef_gap`, `NA_position`, `accepted_datatype`, `...` Arguments pass to the constructor of `aggregator` object. Note that `accepted_datatype` has default value. Downsample with the Largest Triangle Three Buckets (LTTB) aggregation method

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- LTTB_aggregator$new(interleave_gaps = TRUE)
d_agg <- agg$aggregate(
  x = noise_fluct$time, y = noise_fluct$f500, n_out = 1000
  )
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

`min_max_aggregator` *Aggregation using local minimum and maximum values.*

Description

Divide the data into small data ranges and find the maximum and minimum values of each. Note that many samples may be replaced with `NA`, if `interleave_gaps = TRUE` and the original data is increased or decreased monotonically. Use `min_max_ovlp_aggregator` instead in that case. `n_out` must be even number.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::aggregator` -> `min_max_aggregator`

Methods

Public methods:

- `min_max_aggregator$new()`
- `min_max_aggregator$clone()`

Method `new()`**:** Constructor of the Aggregator.

Usage:

```
min_max_aggregator$new(
  ...,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

`interleave_gaps, coef_gap, NA_position, accepted_datatype, ...` Arguments pass to the constructor of `aggregator` object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- min_max_aggregator$new(interleave_gaps = TRUE)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500, 1000)
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

`min_max_ovlp_aggregator` *Aggregation using local minimum and maximum values of which small data ranges have 50% overlaps.*

Description

Divide the data into 50% overlapping intervals and find the maximum and minimum values of each. `n_out` must be even number.

Format

An `R6::R6Class` object

Super class

`shinyHugePlot::aggregator` -> `min_max_ovlp_aggregator`

Methods

Public methods:

- `min_max_ovlp_aggregator$new()`
- `min_max_ovlp_aggregator$clone()`

Method `new()`**:** Constructor of the Aggregator.

Usage:

```
min_max_ovlp_aggregator$new(
  ...,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

`interleave_gaps`, `coef_gap`, `NA_position`, `accepted_datatype`, `...` Arguments pass to the constructor of `aggregator` object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- min_max_ovlp_aggregator$new(interleave_gaps = TRUE)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500, 1000)
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

---

noise_fluct *Time-series fluctuations in sound level*

Description

Results of the measurement of the sound level, where peaks due to road traffic are observed.

Usage

```
noise_fluct
```

Format

A data frame with 32,001 rows and 4 columns:

time time

f500, f1000, f2000 Octave-band sound levels whose center frequencies are 500, 1000 and 2000 Hz.

Author(s)

Junta Tagusari <j.tagusari@eng.hokudai.ac.jp>

---

nth_pnt_aggregator *Aggregation which returns every Nth point.*

Description

Aggregation by extracting every Nth data.

Format

An `R6::R6Class` object

Super class

shinyHugePlot::aggregator -> nth_pnt_aggregator

Methods

Public methods:

- nth_pnt_aggregator$new()
- nth_pnt_aggregator$clone()

Method new(): Constructor of the Aggregator.

Usage:

```
nth_pnt_aggregator$new(
  ...,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

interleave_gaps, coef_gap, NA_position, accepted_datatype, ... Arguments pass to the constructor of aggregator object.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- nth_pnt_aggregator$new(interleave_gaps = TRUE)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500, 1000)
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines")
```

---

null_aggregator *NULL aggregator.*

Description

It does not aggregate the data but returns the full samples within the range.

Format

An R6::R6Class object

Super class

shinyHugePlot::aggregator -> null_aggregator

Methods

Public methods:

- null_aggregator$new()
- null_aggregator$aggregate()
- null_aggregator$clone()

Method new(): Constructor of the Aggregator.

Usage:

```
null_aggregator$new(
  ...,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

interleave_gaps, coef_gap, NA_position, accepted_datatype, ... Arguments pass to the constructor of aggregator object.

Method aggregate(): A function that does nothing other than inserting NAs.

Usage:

```
null_aggregator$aggregate(...)
```

Arguments:

... Arguments passed to super$aggregate.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- null_aggregator$new(interleave_gaps = TRUE)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500)
plotly::plot_ly(
  x = d_agg$x[1:100], y = d_agg$y[1:100], type = "scatter", mode = "lines"
)
```

plotly_build_light *Build* plotly *data with low computation cost*

Description

Before illustrating data using plotly, it must be built (figure$x$data are need to be made using figure$x$attrs). However, because a lot of procedures are necessary, the computation cost is relatively high. With this function, the data is built in quite short time by omitting several procedures for high-frequency data. Note that this function is not universally applicable to all plotly objects but made for high-frequency scatter data. plotly::plotly_build function may return better results in specific cases although it takes more time.

Usage

```
plotly_build_light(fig, vars_hf = c("x", "y", "text", "hovertext"))
```

Arguments

fig	plotly object. Note that fig$x$attrs is not NULL and each fig$x$attrs element has an element named x. This function generates fig$x$data using fig$x$attrs.
vars_hf	Character, optional. Variable names where high frequency data is included. It must include x.

Value

built plotly object

Examples

```
data(noise_fluct)
plotly_build_light(
  plotly::plot_ly(
    x = noise_fluct$time,
    y = noise_fluct$f500,
    name = "level",
    type = "scatter"
  )
)

plotly_build_light(
  plotly::plot_ly(
    data = noise_fluct,
    x = ~time,
    y = ~f500,
    name = "level",
    type = "scatter"
  )
)
```

plotly_datahandler *R6 class for handling plotly data*

Description

A class for handling plotly data, which defines functions used in the downsampler class

Format

An R6::R6Class object

Public fields

figure plotly object.

Active bindings

orig_data Data frame representing plotly traces.

trace_df_default Data frame representing default values of plotly traces. name column represents the names of the attributes. required column represents whether the attributes are necessary to construct a data frame of a trace. data column represents whether the attributes are the data. default attributes represents default values of the attributes. When constructing a data frame of a trace, default values are used if no values are assigned. class column represents the acceptable classes of the attributes.

Methods

Public methods:

- plotly_datahandler$new()
- plotly_datahandler$set_trace_data()
- plotly_datahandler$plotly_data_to_df()
- plotly_datahandler$clone()

Method new(): Constructing an instance. The data contained in a plotly object (figure argument) will be included in the instance (as a reference).

Usage:

```
plotly_datahandler$new(
  figure = NULL,
  legend_options = list(name_prefix = "<b style=\"color:sandybrown\">[S]</b> ",
   name_suffix = "", xdiff_prefix = "<i style=\"color:#fc9944\"> ~", xdiff_suffix =
    "</i>"),
  tz = Sys.timezone(),
  use_light_build = TRUE
)
```

Arguments:

`figure` `plotly` object. The traces of this object will be down-sampled.

`legend_options` Named list, optional. Names of the elements are `name_prefix`, `name_suffix`, `xdiff_prefix`, and `xdiff_suffix`. `name_prefix` and `name_suffix` will be added to the name of the trace when the down-sampling is applied. By default, prefix is a bold orange `[S]` and suffix is none. `xdiff_prefix` and `xdiff_suffix` are employed to show the mean aggregation size of the down-sampling.

`tz` Character, optional. Time zone used to display time-series data. By default `Sys.timezone()`.

`use_light_build` Boolean, optional. Whether `plotly_build_light` is used. It quickly build scatter-type `plotly` data. By default, `TRUE`.

Method `set_trace_data()`**:** In the instance, data is contained as a data frame (see `self$orig_data` for detailed information). Using this method, the data can be added or overwritten. If a data frame (`traces_df` argument) is given, it will be added to `self$orig_data` or reassigned as `self$orig_data`. If attributes to construct a `plotly` object (`...` argument) are given, a data frame is constructed and used.

Usage:

```
plotly_datahandler$set_trace_data(..., traces_df = NULL, append = FALSE)
```

Arguments:

`...` Arguments to constitute a `plotly` attributes, optional. For instance, `x`, `y`, `type`, and `mode` are applicable. See `plotly::plot_ly`.

`traces_df` Data frame, optional. Data frame whose format is agreed with `self$orig_data`. If `traces_df` is given, arguments in `...` are neglected.

`append` Boolean, optional. Whether the data is append or overwrite. By default, `FALSE` (the traces are overwritten).

Method `plotly_data_to_df()`**:** Covert the data contained in `plotly` object to a data frame. A unique id (`uid`) is granted to each data. The data frame will be returned.

Usage:

```
plotly_datahandler$plotly_data_to_df(plotly_data, use_datatable = TRUE)
```

Arguments:

`plotly_data` List. The list whose elements are named list representing `plotly` traces. All elements must have elements named `type`.

`use_datatable` Boolean. If it is `TRUE`, data such as `x` and `y` are nested in a `data.table`, of which key column is `x`. By default, `TRUE`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

range_stat_aggregator *Aggregation which returns the ranges and nominal values within small data ranges*

Description

This aggregator divides the data into no-overlapping intervals and calculate specific statistics that represents the range and nominal values of the data, such as the max, min and mean.

Format

An `R6::R6Class` object

Super classes

`shinyHugePlot::aggregator` -> `shinyHugePlot::rng_aggregator` -> `range_stat_aggregator`

Methods

Public methods:

- `range_stat_aggregator$new()`
- `range_stat_aggregator$clone()`

Method `new()`**:** Constructor of the aggregator.

Usage:

```
range_stat_aggregator$new(
  ...,
  ylwr = min,
  y = mean,
  yupr = max,
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype
)
```

Arguments:

`yupr`, `y`, `ylwr` Functions. Statistical values are calculated using this function. By default, `max`, `mean`, `min`, respectively. Note that the NA values are omitted automatically.

`interleave_gaps`, `coef_gap`, `NA_position`, `accepted_datatype`, `...` Arguments pass to the constructor of `aggregator` object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(noise_fluct)
agg <- range_stat_aggregator$new(
  ylwr = min, y = mean, yupr = max, interleave_gaps = TRUE
)
d_agg <- agg$aggregate(noise_fluct$time, noise_fluct$f500, 100)
plotly::plot_ly(x = d_agg$x, y = d_agg$y, type = "scatter", mode = "lines") %>%
  plotly::add_trace(x = d_agg$x, y = d_agg$ylwr, type = "scatter", mode = "lines")%>%
  plotly::add_trace(x = d_agg$x, y = d_agg$yupr, type = "scatter", mode = "lines")
```

---

rng_aggregator *Aggregation that returns ranges of the data.*

Description

A super class for describing `aggregator` that returns `x`, `y`, `ylwr` and `yupr` values based on given `x` and `y` data.

Format

An `R6::R6Class` object

Value

List of which elements represent the ranges. If there are no `NA`s, the length of the list is 1; multiple lists are obtained if there are `NA`s. Each element of list has `x` and `y` values that surround the range of values.

Super class

`shinyHugePlot::aggregator` -> `rng_aggregator`

Methods

Public methods:

- `rng_aggregator$new()`
- `rng_aggregator$as_plotly_range()`
- `rng_aggregator$as_range()`
- `rng_aggregator$clone()`

Method `new()`**:** Constructor of the Aggregator.

Usage:

```
rng_aggregator$new(
  interleave_gaps,
  coef_gap,
  NA_position,
  accepted_datatype,
  ...
)
```

Arguments:

`interleave_gaps`, `coef_gap`, `NA_position`, `accepted_datatype`, `...` Arguments pass to the constructor of `aggregator` object.

Method `as_plotly_range()`**:** Compute a `plotly` trace to illustrate the range of the data.

Usage:

```
rng_aggregator$as_plotly_range(x, y, ylwr, yupr, opacity = 0.5)
```

Arguments:

`x`, `y`, `ylwr`, `yupr` Outputs of the sub class of `rng_aggregator`.

`opacity` Numeric, optional. Opacity of the range fill. By default, 0.5.

Method `as_range()`**:** Compute `x`, `ylwr` and `yupr` from a `plotly` trace made by `self$as_plotly_range`.

Usage:

```
rng_aggregator$as_range(prng)
```

Arguments:

`prng` List that represents range values, which must contains `x`, `y`. Note that the list may be an element of a list generated by `self$as_plotly_range`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`shinyHugePlot` *shinyHugePlot*

Description

An interactive plot for data with a large sample size using `shiny` and`plotly` can be obtained. For an easy application, see `shiny_hugeplot` function. For a manual application, see `downsampler` class.

shiny_hugeplot *Wrapper for plotting large-sized data using* shinyHugePlot

Description

This is a S3 class function to easily plot large-sized data using downsampler object including plotly and shiny application. Using data that is given as a first argument, shiny application will be constructed and (by default,) executed. As the first argument, many classes are applicable, ranging from a numeric vector representing y values to a downsampler object containing original data, layout of the figure and options for aggregating the original data.

Usage

```
shiny_hugeplot(obj, ...)

## Default S3 method:
shiny_hugeplot(
  obj = NULL,
  y = NULL,
  tz = Sys.timezone(),
  use_light_build = TRUE,
  plotly_options = list(type = "scatter", mode = "lines"),
  plotly_layout_options = list(),
  aggregator = min_max_aggregator$new(),
  n_out = 1000L,
  run_shiny = TRUE,
  downsampler_options = list(),
  shiny_options = list(),
  width = "100%",
  height = "600px",
  ...
)

## S3 method for class 'character'
shiny_hugeplot(
  obj = NULL,
  n_out = 1000L,
  aggregator = min_max_aggregator$new(),
  run_shiny = TRUE,
  use_light_build = TRUE,
  fread_options = list(),
  downsampler_options = list(),
  plotly_options = list(type = "scatter", mode = "lines"),
  plotly_layout_options = list(),
  shiny_options = list(),
  width = "100%",
  height = "600px",
```

```
  ...
)

## S3 method for class 'matrix'
shiny_hugeplot(
  obj = NULL,
  n_out = 1000L,
  aggregator = min_max_aggregator$new(),
  run_shiny = TRUE,
  use_light_build = TRUE,
  downsampler_options = list(),
  plotly_options = list(type = "scatter", mode = "lines"),
  plotly_layout_options = list(),
  shiny_options = list(),
  width = "100%",
  height = "600px",
  ...
)

## S3 method for class 'data.frame'
shiny_hugeplot(
  obj = NULL,
  tz = Sys.timezone(),
  n_out = 1000L,
  aggregator = min_max_aggregator$new(),
  run_shiny = TRUE,
  use_light_build = TRUE,
  downsampler_options = list(),
  plotly_options = list(type = "scatter", mode = "lines"),
  plotly_layout_options = list(),
  shiny_options = list(),
  width = "100%",
  height = "600px",
  ...
)

## S3 method for class 'plotly'
shiny_hugeplot(
  obj,
  n_out = 1000L,
  aggregator = min_max_aggregator$new(),
  run_shiny = TRUE,
  use_light_build = TRUE,
  downsampler_options = list(),
  shiny_options = list(),
  width = "100%",
  height = "600px",
  ...
```

```
)

## S3 method for class 'downsampler'
shiny_hugeplot(
  obj,
  run_shiny = TRUE,
  shiny_options = list(),
  width = "100%",
  height = "600px",
  ...
)
```

Arguments

obj	Numeric/nanotime/POSIXt vector, numeric matrix, data.frame, single character string, plotly object, or downsampler object. If a numeric vector is given, it will be used as y values of the figure of the shiny application (the x values are calculated by seq_along(obj)). It will be interpreted as the x values if you use y argument together. If a nanotime (see nanotime package) vector is given, it will be used as the x values (y argument is mandatory). If a numeric matrix is given, which must have more than 2 columns, the first and second column values will be used as the x and y values. If a data frame is given, which must have columns named x and y, these columns will be used as the x and y values. If a single character string is given, it will be used as a file path to obtain a data frame (data frame will be loaded using data.table::fread). If a plotly object is given, the data and layout of it will be used for constructing the figure of the shiny application. If a downsampler object is given, the data, layout, and down-sampling options for aggregating original data of it will be used for constructing shiny application.
...	Not used.
y	Numeric vector, optional. y values of the figure of shiny application, which is required if the obj argument is used as the x values.
tz	Timezone, optional. It is used to convert the nanotime to the time displayed in the figure. By default, Sys.timezone().
use_light_build	Boolean, optional. Whether shinyHugePlot::plotly_build_light will be used. (if FALSE, plotly::plotly_build will be used) By default, TRUE.
plotly_options	Named list, optional. Arguments passed to plotly::plot_ly.
plotly_layout_options	Named list, optional. Arguments passed to plotly::layout.
aggregator	Instance of R6 classes for aggregating data, optional. The classes can be listed using list_aggregators. By default, min_max_aggregator$new().
n_out	Integer, optional. Number of samples get by the down-sampling. By default, 1000.
run_shiny	Boolean, optional. whether a generated shiny application will be launched. By default, TRUE.

downsampler_options	Named list, optional. Arguments passed to downsampler$new. Note that use aggregator and n_out arguments if you want to set these arguments.
shiny_options	Named list, optional. Arguments passed to shinyApp function.
width, height	Character, optional. Arguments passed to plotlyOutput. By default, 100% and 600px.
fread_options	Named list, optional. Arguments passed to data.table::fread, which is used if a single character string is given as the obj.

Examples

```
data(noise_fluct)

shiny_hugeplot(noise_fluct$f500)
shiny_hugeplot(noise_fluct$time, noise_fluct$f500)
```

---

updatePlotlyH *Function to call a method to update* plotly *traces*

Description

It is used by registering in a shiny application. It receives events in plotly figure and update it using a method of a downsampler instance. See the examples in downsampler class.

Usage

```
updatePlotlyH(
  session,
  outputId,
  relayout_order,
  ds_obj,
  reset = FALSE,
  reload = FALSE
)
```

Arguments

session	session object. The object passed to function given to shinyServer.
outputId	Character. The outputId of the figure, whose data will be down-sampled
relayout_order	Named list. The list generated by plotlyjs_relayout, which is obtained using plotly::event_data.
ds_obj	downsampler instance. The instance containing original data of the figure, which is also used for updating the traces of plotly.

`reset`	Boolean. It it is `TRUE`, the figure will be updated even if `relayout_order` is `NULL`. The ranges of x axes are reset (initialized).
`reload`	Boolean. It it is `TRUE`, the figure will be updated even if `relayout_order` is `NULL`. The ranges of x axes are preserved.

Index