Package ‘InteractiveComplexHeatmap’

May 24, 2024

Type Package

Title Make Interactive Complex Heatmaps

Version 1.13.0

Date 2023-08-10

Depends R (>= 4.0.0), ComplexHeatmap (>= 2.11.0)

Imports grDevices, stats, shiny, grid, GetoptLong, S4Vectors (>= 0.26.1), digest, Ranges, kableExtra (>= 1.3.1), utils, svglite, htmltools, clisymbols, jsonlite, RColorBrewer, fontawesome

Suggests knitr, rmarkdown, testthat, EnrichedHeatmap, GenomicRanges, data.table, circlize, GenomicFeatures, tidyverse, tidyHeatmap, cluster, org.Hs.eg.db, simplifyEnrichment, GO.db, SC3, GOexpress, SingleCellExperiment, scater, gplots, pheatmap, airway, DESeq2, DT, cola, BiocManager, gridtext, HilbertCurve (>= 1.21.1), shinydashboard, SummarizedExperiment, pkgndep, ks

VignetteBuilder knitr

Description This package can easily make heatmaps which are produced by the ComplexHeatmap package into interactive applications. It provides two types of interactivities:
1. on the interactive graphics device, and 2. on a Shiny app. It also provides functions for integrating the interactive heatmap widgets for more complex Shiny app development.

biocViews Software, Visualization, Sequencing

URL https://github.com/jokergoo/InteractiveComplexHeatmap

BugReports https://github.com/jokergoo/InteractiveComplexHeatmap/issues

License MIT + file LICENSE

git_url https://git.bioconductor.org/packages/InteractiveComplexHeatmap
git_branch devel
git_last_commit bd8471f
git_last_commit_date 2024-04-30
### Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_column_indices</td>
<td>2</td>
</tr>
<tr>
<td>all_row_indices</td>
<td>3</td>
</tr>
<tr>
<td>getPositionFromBrush</td>
<td>3</td>
</tr>
<tr>
<td>getPositionFromClick</td>
<td>4</td>
</tr>
<tr>
<td>getPositionFromDblclick</td>
<td>5</td>
</tr>
<tr>
<td>getPositionFromHover</td>
<td>5</td>
</tr>
<tr>
<td>HeatmapInfoOutput</td>
<td>6</td>
</tr>
<tr>
<td>htPositionsOnDevice</td>
<td>7</td>
</tr>
<tr>
<td>htShiny</td>
<td>8</td>
</tr>
<tr>
<td>htShinyExample</td>
<td>11</td>
</tr>
<tr>
<td>ht_shiny</td>
<td>11</td>
</tr>
<tr>
<td>interactactivate</td>
<td>12</td>
</tr>
<tr>
<td>interactivate.DESeqDataSet</td>
<td>13</td>
</tr>
<tr>
<td>interactivate.kde</td>
<td>13</td>
</tr>
<tr>
<td>interactivateDensity2D</td>
<td>14</td>
</tr>
<tr>
<td>InteractiveComplexHeatmapModal</td>
<td>15</td>
</tr>
<tr>
<td>InteractiveComplexHeatmapOutput</td>
<td>17</td>
</tr>
<tr>
<td>InteractiveComplexHeatmapWidget</td>
<td>19</td>
</tr>
<tr>
<td>is_in_sub_heatmap</td>
<td>21</td>
</tr>
<tr>
<td>makeInteractiveComplexHeatmap</td>
<td>21</td>
</tr>
<tr>
<td>originalHeatmapOutput</td>
<td>23</td>
</tr>
<tr>
<td>rand_mat</td>
<td>24</td>
</tr>
<tr>
<td>record_observation</td>
<td>25</td>
</tr>
<tr>
<td>selectArea</td>
<td>26</td>
</tr>
<tr>
<td>selectPosition</td>
<td>27</td>
</tr>
<tr>
<td>subHeatmapOutput</td>
<td>28</td>
</tr>
</tbody>
</table>

### Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_column_indices</td>
<td>29</td>
</tr>
</tbody>
</table>

---

#### Description

Get all column indices from the selected data frame

#### Usage

```r
all_column_indices(df)
```
**all_row_indices**

**Arguments**

- `df` The selected data frame.

**Examples**

```r
# There is no example
NULL
```

---

**getPositionFromBrush**

Get the position of the brushed area on the heatmap image

**Description**

Get the position of the brushed area on the heatmap image

**Usage**

```r
g getPositionFromBrush(brush, ratio = 1)
```

**Arguments**

- `brush` The input brush object. Assume `heatmap_brush` is the ID set to argument `brush` in `plotOutput`, then the value here is `input$heatmap_brush`.
- `ratio` The relative resolution. The value should the ratio between `res` set in `makeInteractiveComplexHeatmap` and 72 (`res/72`).
getPositionFromClick

Value

A list of length two. The two elements correspond to the coordinates of the two diagonal points.

See Also

g getPositionFromClick, getPositionFromHover, getPositionFromDblclick.

Examples

# There is no example
NULL

getPositionFromClick  Get the position of clicked point on the heatmap image

Description

Get the position of clicked point on the heatmap image

Usage

g getPositionFromClick(click, ratio = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click</td>
<td>The input click object. Assume heatmap_click is the ID set to argument click in <code>plotOutput</code>, then the value here is <code>input$heatmap_click</code>.</td>
</tr>
<tr>
<td>ratio</td>
<td>The relative resolution. The value should the ratio between <code>res</code> set in <code>makeInteractiveComplexHeatmap</code> and 72 (<code>res/72</code>).</td>
</tr>
</tbody>
</table>

Value

A unit object of length two which are the coordinates of the clicked points.

See Also

g getPositionFromBrush, getPositionFromHover, getPositionFromDblclick.

Examples

# There is no example
NULL
getPositionFromDblclick

Get the position of double clicked point on the heatmap image

Description
Get the position of double clicked point on the heatmap image

Usage
getPositionFromDblclick(dblclick, ratio = 1)

Arguments
- **dblclick**: The input dblclick object. Assume heatmap_dblclick is the ID set to argument dblclick in plotOutput, then the value here is input$heatmap_dblclick.
- **ratio**: The relative resolution. The value should the ratio between res set in makeInteractiveComplexHeatmap and 72 (res/72).

Value
A unit object of length two which are the coordinates of the double clicked points.

Examples
```r
# There is no example
NULL
```

getPositionFromHover

Get the position of hovered point on the heatmap image

Description
Get the position of hovered point on the heatmap image

Usage
getPositionFromHover(hover, ratio = 1)

Arguments
- **hover**: The input hover object. Assume heatmap_hover is the ID set to argument hover in plotOutput, then the value here is input$heatmap_hover.
- **ratio**: The relative resolution. The value should the ratio between res set in makeInteractiveComplexHeatmap and 72 (res/72).
Value

A `unit` object of length two which are the coordinates of the hover points.

Examples

```r
# There is no example
NULL
```

---

HeatmapInfoOutput  
\textit{UI for the output}

---

Description

UI for the output

Usage

```r
HeatmapInfoOutput(heatmap_id, title = NULL, width = 400, 
output_ui = default_output_ui(heatmap_id),
output_ui_float = FALSE, action = NULL, response = NULL, internal = FALSE)
```

Arguments

- `heatmap_id`: ID of the plot.
- `title`: Title of the output.
- `width`: Width of the output div.
- `output_ui`: A `htmlOutput` or other *Output object (defined in shiny or other related packages).
- `output_ui_float`: Whether the UI defined by `output_ui` floats at the mouse positions.
- `action`: It is only used when `output_ui_float = TRUE` to properly bind the floating frame to the event on heatmap (i.e. click, hover or dblclick). If `HeatmapInfoOutput` is executed after `originalHeatmapOutput`, the value for it is automatically decided.
- `response`: It is only used when `output_ui_float = TRUE` and `response = "brush"` or `response = "brush-output"`, so that single clicking or hovering won’t have any effect, in other word, there is only response from brushing. If `HeatmapInfoOutput` is executed after `originalHeatmapOutput`, the value for it is automatically decided.
- `internal`: Internally used.

See Also

`originalHeatmapOutput`, `subHeatmapOutput`. 
htPositionsOnDevice

Examples

# See examples on the help page of originalHeatmapOutput()

---

Get heatmap positions on the graphics device

Usage

htPositionsOnDevice(ht_list = get_last_ht(), unit = "inch", valueOnly = FALSE, include_annotation = FALSE, calibrate = TRUE)

Arguments

- **ht_list**: A `HeatmapList-class` object returned by `draw,Heatmap-method` or `draw,HeatmapList-method`. If it is omitted, it uses the last generated heatmap.
- **unit**: The unit.
- **valueOnly**: Whether only return the numeric values.
- **include_annotation**: Internally used.
- **calibrate**: Internally used.

Details

ht_list must have been already updated by `draw()` function. The function needs to be executed under a graphics device where the heatmap is written.

Value

It returns a `DataFrame` object of the position of every heatmap slice.

Examples

```r
if(dev.interactive()) {
  m = matrix(rnorm(100), 10)
  ht = Heatmap(m, row_km = 2, column_km = 2)
  ht = draw(ht)
  pos = htPositionsOnDevice(ht)

  InteractiveComplexHeatmap:::redraw_ht_vp(pos)
}
```
Interactive heatmaps as a Shiny app

**Description**

Interactive heatmaps as a Shiny app

**Usage**

```r
htShiny(ht_list = get_last_ht(), title = NULL,
        description = NULL, hline = TRUE, html = NULL,
        # parameters passed to InteractiveComplexHeatmapOutput()
        heatmap_id = NULL, title1 = "Original heatmap", title2 = "Selected sub-heatmap",
        width1 = ifelse(layout == "1|(2-3)", 800, 450),
        height1 = ifelse(layout == "1-(2|3)", 700, 350),
        width2 = 400,
        height2 = 350,
        width3 = ifelse(layout == "(1-2)|3", 800, 400),
        layout = ifelse("brush" %in% response, "(1-2)|3", "1-3"), compact = FALSE,
        action = "click", cursor = TRUE, response = c(action, "brush"),
        brush_opt = list(stroke = "#f00", opacity = 0.6),
        output_ui_float = FALSE,
        # specific for sub-heatmap
        show_cell_fun = TRUE, show_layer_fun = TRUE,
        save = NULL, app_options = list())
```

**Arguments**

- **ht_list**: A `Heatmap-class` or a `HeatmapList-class` object. If it is not specified, the last generated heatmap is used. The heatmap object should better be already updated by `draw()` function.
- **title**: Title of the app.
- **description**: Description of the app. The content will be wrapped by a `p` tag and inserted before the interactive heatmap widget.
- **hline**: Whether to add the horizontal line (by `hr` tag) after `description`.
- **html**: HTML fragment inserted below the heatmap. The value can be a string or be wrapped by `HTML`.
- **heatmap_id**: Pass to `InteractiveComplexHeatmapOutput`.
- **title1**: Pass to `InteractiveComplexHeatmapOutput`.
- **title2**: Pass to `InteractiveComplexHeatmapOutput`.
- **width1**: Pass to `InteractiveComplexHeatmapOutput`.
htShiny

**height1**
Pass to `InteractiveComplexHeatmapOutput`.

**width2**
Pass to `InteractiveComplexHeatmapOutput`.

**height2**
Pass to `InteractiveComplexHeatmapOutput`.

**width3**
Pass to `InteractiveComplexHeatmapOutput`.

**layout**
Pass to `InteractiveComplexHeatmapOutput`.

**compact**
Pass to `InteractiveComplexHeatmapOutput`.

**action**
Pass to `InteractiveComplexHeatmapOutput`.

**cursor**
Pass to `InteractiveComplexHeatmapOutput`.

**response**
Pass to `InteractiveComplexHeatmapOutput`.

**brush_opt**
Pass to `InteractiveComplexHeatmapOutput`.

**output_ui_float**
Pass to `InteractiveComplexHeatmapOutput`.

**show_cell_fun**
Whether show graphics made by `cell_fun` on the main heatmap?

**show_layer_fun**
Whether show graphics made by `cell_fun` on the main heatmap?

**save**
The value can be set to a folder name so that the shiny app is saved into several files.

**app_options**
All pass to the options argument in `shinyApp`.

**Details**

With any `Heatmap/HeatmapList` object, directly send to `htShiny()` to create a Shiny app for the heatmap(s):

```
htShiny(ht_list)
```

If the heatmaps are already drawn, `ht_list` can be omitted and the last heatmap object is retrieved automatically:

```
Heatmap(...) + other_heatmaps_or_annotations # or other functions that internally use Heatmap()
htShiny()
```

**Value**

A Shiny app object.

**See Also**

- [https://jokergoo.shinyapps.io/interactive_complexheatmap/](https://jokergoo.shinyapps.io/interactive_complexheatmap/)
- [https://jokergoo.shinyapps.io/interactive_complexheatmap_vertical/](https://jokergoo.shinyapps.io/interactive_complexheatmap_vertical/)
- [https://jokergoo.shinyapps.io/interactive_densityheatmap/](https://jokergoo.shinyapps.io/interactive_densityheatmap/)
- [https://jokergoo.shinyapps.io/interactive_oncoprint/](https://jokergoo.shinyapps.io/interactive_oncoprint/)
- [https://jokergoo.shinyapps.io/interactive_enrichedheatmap/](https://jokergoo.shinyapps.io/interactive_enrichedheatmap/)
- [https://jokergoo.shinyapps.io/interactive_upsetp/](https://jokergoo.shinyapps.io/interactive_upsetp/)
There are also many examples that can be get with `htShinyExample`.

**Examples**

```r
# use last generated heatmap
if(interactive() && dev.interactive()) {
  m = matrix(rnorm(100), 10)
  Heatmap(m)
  htShiny()
}

# by providing a heatmap/heatmap list
if(interactive()) {
  m = matrix(rnorm(100), 10)
  rownames(m) = 1:10
  colnames(m) = 1:10

  ht = Heatmap(m)
  ht = draw(ht)
  htShiny(ht)
}

# vertical heatmap list
if(interactive()) {
  m1 = matrix(rnorm(100), 10)
  rownames(m1) = 1:10
  colnames(m1) = 1:10
  ht1 = Heatmap(m1, row_km = 2, column_km = 2)

  m2 = matrix(sample(letters[1:10], 100, replace = TRUE), 10)
  ht2 = Heatmap(m2)

  ht_list = draw(ht1 + ht2)
  htShiny(ht_list)

  ht_list = ht1 %v% ht2
  htShiny(ht_list)
}

# compact mode
if(interactive()) {
  m = matrix(rnorm(100), 10)
  Heatmap(m)
  htShiny(compact = TRUE)
}
```
htShinyExample

Examples of interactive complex heatmaps

Description
Examples of interactive complex heatmaps

Usage
htShinyExample(which)

Arguments
which
An index of which example to use. The list of all examples can be obtained by executing `htShinyExample` with no argument.

Details
In every example, there is a Shiny app opened, which also includes source code that generates this app.

Value
A Shiny app object.

Examples

```r
# list all examples
htShinyExample()

if(interactive()) {
  htShinyExample(4.2)
}
```

ht_shiny
Interactive heatmaps as a Shiny app

Description
Interactive heatmaps as a Shiny app

Usage
ht_shiny(...)
interactivate

Arguments

... All goes to \texttt{htShiny}.

Value

A Shiny app object.

Examples

# There is no example
\texttt{NULL}

\begin{Verbatim}
interactivate \hspace{1cm} \textit{Generic function for interactivate an object in an interactive Shiny app}
\end{Verbatim}

Description

Generic function for interactivate an object in an interactive Shiny app

Usage

\texttt{interactivate(x, \ldots)}

Arguments

x An object.

\ldots Other arguments.

Examples

# There is no example
\texttt{NULL}
interactivate.DESeqDataSet

Visualize DESeq2 result in an interactive Shiny app

Description

Visualize DESeq2 result in an interactive Shiny app

Usage

```r
## S3 method for class 'DESeqDataSet'
interactivate(x, res = DESeq2::results(x), seed = 123, ...)
```

Arguments

- `x`: A `DESeqDataSet` class object. It is normally returned by `DESeq`.
- `res`: The object returned by `results`.
- `seed`: Random seed. It is mainly set for the random colors of annotations.
- `...`: Other arguments.

Examples

```r
if(interactive()) {
  require(airway)
  data(airway)
  se = airway

  require(DESeq2)
  dds = DESeqDataSet(se, design = ~ dex)
  keep = rowSums(counts(dds)) >= 10
  dds = dds[keep, ]
  dds$dex = relevel(dds$dex, ref = "untrt")
  dds = DESeq(dds)

  interactivate(dds)
}
```

interactivate.kde

Interactive Shiny application for 2D density distribution

Description

Interactive Shiny application for 2D density distribution
Usage

## S3 method for class 'kde'
interactivate(x, ...)

Arguments

x  
a kde object generated by kde.

...  
Other arguments.

Examples

if(interactive()) {
  require(ks)
  lt = readRDS(system.file("extdata", "2d_density_xy.rds", package = "InteractiveComplexHeatmap"))
  data = cbind(lt$x, lt$y)
  fit = kde(data)
  interactivate(fit)
}

interactivateDensity2D

Interactive Shiny application for 2D density distribution

Description

Interactive Shiny application for 2D density distribution

Usage

interactivateDensity2D(x, y, ...)

Arguments

x  
A numeric vector.

y  
A numeric vector.

...  
All pass to kde.

Examples

if(interactive()) {
  lt = readRDS(system.file("extdata", "2d_density_xy.rds", package = "InteractiveComplexHeatmap"))
  interactivateDensity2D(lt$x, lt$y)
}
InteractiveComplexHeatmapModal

Interactive complex heatmap modal dialog

Description

Interactive complex heatmap modal dialog

Usage

InteractiveComplexHeatmapModal(
  input, output, session, ht_list, heatmap_id = NULL,
  # parameters passed to InteractiveComplexHeatmapOutput()
  title1 = "Original heatmap", title2 = "Selected sub-heatmap",
  width1 = ifelse(layout == "1|2-3", 800, 450),
  height1 = ifelse(layout == "1-(2|3)", 700, 350),
  width2 = 370,
  height2 = 350,
  width3 = ifelse(layout == "(1-2)|3", 800, 370),
  layout = ifelse("brush" %in% response, "(1-2)|3", "1-3"), compact = FALSE,
  action = "click", cursor = TRUE, response = c(action, "brush"),
  brush_opt = list(stroke = "#f00", opacity = 0.6),
  output_ui = TRUE, output_ui_float = FALSE,
  # parameters passed to makeInteractiveComplexHeatmap()
  click_action = NULL, brush_action = NULL,
  # other configurations
  js_code = "", close_button = TRUE, cancel_action = c("remove", "hide"))

Arguments

input          Passed from the Shiny server function.
output         Passed from the Shiny server function.
session        Passed from the Shiny server function.
ht_list        A Heatmap-class or a HeatmapList-class object.
heatmap_id     ID of the plot. If it is not specified, an internal ID is assigned.
title1         Pass to InteractiveComplexHeatmapOutput.
title2         Pass to InteractiveComplexHeatmapOutput.
width1         Pass to InteractiveComplexHeatmapOutput.
height1        Pass to InteractiveComplexHeatmapOutput.
width2         Pass to InteractiveComplexHeatmapOutput.
height2        Pass to InteractiveComplexHeatmapOutput.
width3 Pass to InteractiveComplexHeatmapOutput.
layout Pass to InteractiveComplexHeatmapOutput.
compact Pass to InteractiveComplexHeatmapOutput.
action Pass to InteractiveComplexHeatmapOutput.
cursor Pass to InteractiveComplexHeatmapOutput.
response Pass to InteractiveComplexHeatmapOutput.
brush_opt Pass to InteractiveComplexHeatmapOutput.
output_ui Pass to InteractiveComplexHeatmapOutput.
output_ui_float Pass to InteractiveComplexHeatmapOutput.
click_action Pass to makeInteractiveComplexHeatmap.
brush_action Pass to makeInteractiveComplexHeatmap.
js_code Additional JavaScript code that is put after the interactive heatmap UI. The value can be a text or a function that takes "heatmap ID" as the argument and returns the formatted JavaScript code.
close_button Whether to add a close button at the end of the widget. If it is FALSE, the widget can be closed by clicking outside of the widget.
cancel_action Whether to remove the UI from HTML or just hide it when the UI is closed.

Details
It creates an interactive heatmap "modal dialog" according to a certain action.
The function is normally put inside observe or observeEvent.

Value
No value is returned.

Examples
if(interactive()) {
  require(ComplexHeatmap)

  ui = fluidPage(
    actionButton("show_heatmap", "Generate_heatmap"),
  )

  server = function(input, output, session) {
    m = matrix(rnorm(100), 10)
    ht = Heatmap(m)

    observeEvent(input$show_heatmap, {
      InteractiveComplexHeatmapModal(input, output, session, ht)
    })
  }
  shiny::shinyApp(ui, server)
}
InteractiveComplexHeatmapOutput

UI for the interactive complex heatmaps

Description

UI for the interactive complex heatmaps

Usage

InteractiveComplexHeatmapOutput(heatmap_id = NULL,
                              title1 = "Original heatmap", title2 = "Selected sub-heatmap",
                              title3 = if(output_ui_float) NULL else "Output",
                              width1 = ifelse(layout == "1|(2-3)", 800, 450),
                              height1 = ifelse(layout == "1-(2|3)", 700, 350),
                              width2 = 400,
                              height2 = 350,
                              width3 = NULL,
                              layout = ifelse("brush" %in% response, "(1-2)|3", "1-3"),
                              compact = FALSE,
                              action = "click", cursor = TRUE,
                              response = c(action, "brush"),
                              brush_opt = list(stroke = "#f00", opacity = 0.6),
                              output_ui = default_output_ui(heatmap_id),
                              output_ui_float = FALSE, containment = FALSE,
                              internal = FALSE,
                              ...
)

Arguments

heatmap_id  ID of the plot. If it is not specified, an internal ID is assigned.
title1     Title of the original heatmap.
title2     Title of the sub-heatmap.
title3     Title of the output.
width1     Width of the original heatmap.
height1    Height of the original heatmap.
width2     Width of the sub-heatmap.
height2    Height of the sub-heatmap.
width3     Width of the output div.
layout     One of "(1|2)-3", "1-(2|3)", "1-2-3", "1|2|3", "1|(2-3)". If brush is not set with the argument response, which means there is no sub-heatmap panel, the code 2 can be omitted.
compact   If the value is TRUE, there will be no sub-heatmap, and output floats at the mouse position when click/hover on the original heatmap.
**InteractiveComplexHeatmapOutput**

- **action**: Which action for selecting single cells on the heatmap? Value should be click, hover or dblclick.
- **cursor**: When moving mouse on heatmap, whether to show the cursors on the four sides?
- **response**: Which action needs to be responded on the server side? Value should be in click/hover/dblclick, brush and brush-output. brush responds in two places which are the sub-heatmap and the output components and brush-output only responds in the output component.
- **brush_opt**: A list of parameters passed to brushOpt. Do not set an ID for the brush. An internal brush ID is automatically set.
- **output_ui**: A `htmlOutput` or other `*Output` object (defined in shiny or other related packages). If it is set to NULL, there is no output component in the app.
- **output_ui_float**: Whether the UI defined by `output_ui` floats at the mouse positions.
- **containment**: Whether the resizing is restricted in a certain parent div? Value can be TRUE/FALSE or a JQuery selector.
- **internal**: Internally used.
- **...**: Pass to the UI container which is wrapped by `fluidPage`.

**Details**

This function generates HTML fragment for the interactive UI. See the example in `makeInteractiveComplexHeatmap` page.

`layout` is defined as follows (1 for the original heatmap, 2 for the selected sub-heatmap and 3 is for the output:

- "(1-2)|3": Heatmap and sub-heatmap are in a same row, and output is in a second row. This is the default layout.
- "1|(2-3)": Heatmap is in a single row, while sub-heatmap and output are in a second row.
- "1-2-3": All three components are in a same row.
- "1|2|3": Each component is in a single row.
- "1-(2|3)": Being different from the other four layouts, this is a two-column layout. Heatmap is in a single column. Sub-heatmap and output are vertically aligned and the two are in the second column.

The hover event is implemented with [https://github.com/websanova/mousestop](https://github.com/websanova/mousestop).

**Value**

A UI that can be used in Shiny.

**Examples**

```r
# There is no example
NULL
```
InteractiveComplexHeatmapWidget

Interactive complex heatmap widget

Description

Interactive complex heatmap widget

Usage

InteractiveComplexHeatmapWidget(
  input, output, session, ht_list, heatmap_id = NULL, output_id,

  # parameters passed to InteractiveComplexHeatmapOutput()
  title1 = "Original heatmap", title2 = "Selected sub-heatmap",
  width1 = ifelse(layout == "1|2-3", 800, 450),
  height1 = ifelse(layout == "1-(2|3)", 700, 350),
  width2 = 370,
  height2 = 350,
  width3 = ifelse(layout == "(1-2)|3", 800, 370),
  layout = ifelse("brush" %in% response, "(1-2)|3", "1-3"), compact = FALSE,
  action = "click", cursor = TRUE, response = c(action, "brush"),
  brush_opt = list(stroke = "#f00", opacity = 0.6),
  output_ui = TRUE, output_ui_float = FALSE,

  # other configurations
  js_code = ", close_button = TRUE, cancel_action = c("remove", "hide")
)

Arguments

input Passed from the Shiny server function.
output Passed from the Shiny server function.
session Passed from the Shiny server function.
ht_list A Heatmap-class or a HeatmapList-class object.
heatmap_id ID of the plot. If it is not specified, an internal ID is assigned.
output_id Where the heatmap is put.
title1 Pass to InteractiveComplexHeatmapOutput.
title2 Pass to InteractiveComplexHeatmapOutput.
width1 Pass to InteractiveComplexHeatmapOutput.
height1 Pass to InteractiveComplexHeatmapOutput.
width2 Pass to InteractiveComplexHeatmapOutput.
InteractiveComplexHeatmapWidget

**Parameters**

- `height2`: Pass to `InteractiveComplexHeatmapOutput`.
- `width3`: Pass to `InteractiveComplexHeatmapOutput`.
- `layout`: Pass to `InteractiveComplexHeatmapOutput`.
- `compact`: Pass to `InteractiveComplexHeatmapOutput`.
- `action`: Pass to `InteractiveComplexHeatmapOutput`.
- `cursor`: Pass to `InteractiveComplexHeatmapOutput`.
- `response`: Pass to `InteractiveComplexHeatmapOutput`.
- `brush_opt`: Pass to `InteractiveComplexHeatmapOutput`.
- `output_ui`: Pass to `InteractiveComplexHeatmapOutput`.
- `output_ui_float`: Pass to `InteractiveComplexHeatmapOutput`.
- `click_action`: Pass to `makeInteractiveComplexHeatmap`.
- `brush_action`: Pass to `makeInteractiveComplexHeatmap`.
- `js_code`: Additional JavaScript code that is put after the interactive heatmap UI. The value can be a text or a function that takes "heatmap ID" as the argument and returns the formatted JavaScript code.
- `close_button`: Whether to add a close button at the end of the widget.
- `cancel_action`: Whether to remove the UI from HTML or just hide it when the UI is closed.

**Details**

It creates an interactive heatmap widget according to a certain action. The UI is placed to the output ID that user defined.

The function is normally put inside `observe` or `observeEvent`.

**Value**

No value is returned.

**Examples**

```r
if(interactive()) {
  require(ComplexHeatmap)

  ui = fluidPage(
    actionButton("show_heatmap", "Generate_heatmap"),
    htmlOutput("heatmap_output")
  )

  server = function(input, output, session) {
    m = matrix(rnorm(100), 10)
    ht = Heatmap(m)

    observeEvent(input$show_heatmap, {
      InteractiveComplexHeatmapWidget(input, output, session, ht, output_id = "heatmap_output")
    })
  }
```

is_in_sub_heatmap

Test whether it is in sub heatmap

Description
Test whether it is in sub heatmap

Usage
is_in_sub_heatmap()

Details
Normally, it is used in cell_fun/layer_fun.

Examples
# There is no example
NULL

makeInteractiveComplexHeatmap
Process heatmaps on the sever side

Description
Process heatmaps on the sever side

Usage
makeInteractiveComplexHeatmap(input, output, session, ht_list,
   heatmap_id = shiny_env$current_heatmap_id,
   click_action = NULL, hover_action = NULL,
   dblclick_action = NULL, brush_action = NULL, res = 72,
   show_cell_fun = TRUE, show_layer_fun = TRUE)
makeInteractiveComplexHeatmap

Arguments

- **input**: Passed from the Shiny server function.
- **output**: Passed from the Shiny server function.
- **session**: Passed from the Shiny server function.
- **ht_list**: A `Heatmap-class` or a `HeatmapList-class` object.
- **heatmap_id**: The corresponding heatmap ID from the UI. If there is only one interactive heatmap in the app, this argument does not need to be specified and it will use the current one used in `InteractiveComplexHeatmapOutput`.
- **click_action**: Additional actions on the server side when receiving a click event on the UI. This self-defined function should accept two or four arguments. If it is two arguments, they should be `df` and `output` and if it is four arguments, they should be `df`, `input`, `output` and `session`.
- **hover_action**: Additional actions at the server side when receiving a hover event on the UI.
- **dblclick_action**: Additional actions at the server side when receiving a dblclick event on the UI.
- **brush_action**: Additional actions at the server side when receiving a brush event on the UI.
- **res**: Resolution of the plot, pass to `renderPlot`.
- **show_cell_fun**: Whether show graphics made by `cell_fun` on the main heatmap?
- **show_layer_fun**: Whether show graphics made by `cell_fun` on the main heatmap?

Value

No value is returned.

Examples

```r
if(interactive()) {
  ht = Heatmap(m)
  ht = draw(ht)

  ui = fluidPage(
    InteractiveComplexHeatmapOutput()
  )

  server = function(input, output, session) {
    makeInteractiveComplexHeatmap(input, output, session, ht)
  }

  shiny::shinyApp(ui, server)
}
```
originalHeatmapOutput  UI for the original heatmap

Description

UI for the original heatmap

Usage

originalHeatmapOutput(heatmap_id, title = NULL, width = 450, height = 350, action = "click", cursor = TRUE, response = c(action, "brush"), brush_opt = list(stroke = "#f00", opacity = 0.6), containment = FALSE, internal = FALSE)

Arguments

heatmap_id  ID of the plot.
title  Title of the original heatmap.
width  Width of the original heatmap.
height  Height of the original heatmap.
action  Which action for selecting single cells on the heatmap? Value should be click, hover or dblclick.
cursor  When moving mouse on heatmap, whether to show the cursors on the four sides?
response  Which action needs to be responded on the server side? Value should be in click/hover/dblclick, brush and brush-output. brush responds in two places which are the sub-heatmap and the output components and brush-output only responds in the output component.
brush_opt  A list of parameters passed to brushOpt. Do not set an ID for the brush. An internal brush ID is automatically set.
containment  Whether the resizing is restricted in a certain parent div? Value can be TRUE/FALSE or a JQuery selector.
internal  Internally used.

See Also

subHeatmapOutput, HeatmapInfoOutput.
Examples

```r
if(interactive()) {
  require(shinydashboard)
  m = matrix(rnorm(100), 10)
  ht = Heatmap(m)

  body = dashboardBody(
    fluidRow(
      box(title = "Original heatmap", width = 4, solidHeader = TRUE, status = "primary",
          originalHeatmapOutput("ht")
      ),
      box(title = "Sub-heatmap", width = 4, solidHeader = TRUE, status = "primary",
          subHeatmapOutput("ht")
      ),
      box(title = "Output", width = 4, solidHeader = TRUE, status = "primary",
          HeatmapInfoOutput("ht")
      )
    )
  )
  ui = dashboardPage(
    dashboardHeader(),
    dashboardSidebar(),
    body
  )
  server = function(input, output, session) {
    makeInteractiveComplexHeatmap(input, output, session, ht, "ht")
  }
  shinyApp(ui, server)
}
```

---

**rand_mat**

A random matrix

**Description**

A random matrix

**Usage**

```r
data(rand_mat)
```

**Details**

Following code was used to generate `rand_mat`:

```r
set.seed(123)
rand_mat = cbind(rbind(matrix(rnorm(20*20, mean = 1, sd = 0.5), nr = 20),
                       matrix(rnorm(20*20, mean = 0, sd = 0.5), nr = 20),
                       matrix(rnorm(20*20, mean = 0, sd = 0.5), nr = 20)),
               matrix(rnorm(20*20, mean = 0, sd = 0.5), nr = 20))
```
record_observation

```r
rbind(matrix(rnorm(20*20, mean = 0, sd = 0.5), nr = 20),
      matrix(rnorm(20*20, mean = 1, sd = 0.5), nr = 20),
      matrix(rnorm(20*20, mean = 0, sd = 0.5), nr = 20)),
```

```r
rbind(matrix(rnorm(20*20, mean = 0.5, sd = 0.5), nr = 20),
      matrix(rnorm(20*20, mean = 0.5, sd = 0.5), nr = 20),
      matrix(rnorm(20*20, mean = 1, sd = 0.5), nr = 20))
```

```r
) + matrix(rnorm(60*60, sd = 0.5), nr = 60)
```

```r
colnames(rand_mat) = paste0("C", 1:60)
rownames(rand_mat) = paste0("R", 1:60)
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```r
data(rand_mat)
rand_mat
```

Description

Record the observation object

Usage

```r
record_observation(obs, heatmap_id = shiny_env$current_heatmap_id)
```

Arguments

- `obs`: Observation object returned by `observe` or `observeEvent`.
- `heatmap_id`: The Heatmap ID.

Examples

```r
# There is no example
NULL
```
selectArea

Select an area in the heatmap

Description

Select an area in the heatmap

Usage

selectArea(ht_list = get_last_ht(), pos1 = NULL, pos2 = NULL, mark = TRUE, verbose = TRUE,
           ht_pos = NULL, include_annotation = FALSE, calibrate = TRUE)

Arguments

- **ht_list**: A `HeatmapList-class` object returned by `draw,Heatmap-method` or `draw,HeatmapList-method`. If it is omitted, it uses the last generated heatmap.
- **mark**: Whether to mark the selected area as a rectangle.
- **pos1**: If the value is `NULL`, it can be selected by click on the heatmap (of course, the heatmap should be on the interactive graphics device). If it is set, it must be a `unit` object with length two which corresponds to the x and y position of the point.
- **pos2**: Another point as `pos1`, together with `pos1` defines the selected region.
- **verbose**: Whether to print messages.
- **ht_pos**: A value returned by `htPositionsOnDevice`.
- **include_annotation**: Internally used.
- **calibrate**: Internally used. Mainly works for Rstudio desktop IDE.

Details

The regions can be selected interactively or selected manually by setting `pos1` and `pos2`.

Value

A `DataFrame` object with row indices and column indices corresponding to the selected region.

Examples

```r
if(dev.interactive()) {
  m = matrix(rnorm(100), 10)
  rownames(m) = 1:10
  colnames(m) = 1:10

  ht = Heatmap(m)
  ht = draw(ht)
  selectArea(ht)
} 
```
selectPosition

set.seed(123)
ht = Heatmap(m, row_km = 2, column_km = 2)
ht = draw(ht)
selectArea(ht)
}

selectPosition  Select a position in the heatmap

Description

Select a position in the heatmap

Usage

selectPosition(ht_list = get_last_ht(), pos = NULL, mark = TRUE, verbose = TRUE, ht_pos = NULL, calibrate = TRUE)

Arguments

ht_list  A HeatmapList-class object returned by draw,Heatmap-method or draw,HeatmapList-method. If it is omitted, it uses the last generated heatmap.
mark  Whether to mark the selected position as a point.
pos  If the value is NULL, it can be selected by click on the heatmap (of course, the heatmap should be on the interactive graphics device). If it is set, it must be a unit object with length two which corresponds to the x and y position of the point.
verbose  Whether to print messages.
ht_pos  A value returned by htPositionsOnDevice.
calibrate  Internally used. Mainly works for Rstudio desktop IDE.

Details

The regions can be selected interactively or selected manually by setting pos.

Value

A DataFrame object with row indices and column indices corresponding to the selected position.
Examples

```r
if(dev.interactive()) {
  m = matrix(rnorm(100), 10)
  rownames(m) = 1:10
  colnames(m) = 1:10

  ht = Heatmap(m)
  ht = draw(ht)
  selectPosition(ht)
}
```

---

**subHeatmapOutput**  
*UI for the sub-heatmaps*

**Description**

UI for the sub-heatmaps

**Usage**

```r
subHeatmapOutput(heatmap_id, title = NULL, width = 400, height = 350, containment = FALSE, internal = FALSE)
```

**Arguments**

- `heatmap_id`  
  ID of the plot.
- `title`  
  Title of the sub-heatmap.
- `width`  
  Width of the sub-heatmap.
- `height`  
  Height of the sub-heatmap.
- `containment`  
  Whether the resizing is restricted in a certain parent div? Value can be TRUE/FALSE or a JQuery selector.
- `internal`  
  Internally used.

**See Also**

`originalHeatmapOutput`.

**Examples**

```r
# See examples on the help page of originalHeatmapOutput()
```
Index

all_column_indices, 2
all_row_indices, 3
brushOpts, 18, 23
DataFrame, 7, 26, 27
DESeq, 13
DESeqDataSet, 13
fluidPage, 18
getPositionFromBrush, 3, 4
getPositionFromClick, 4, 4
getPositionFromDblclick, 4, 5
getPositionFromHover, 4, 5
HeatmapInfoOutput, 6, 6, 23
ht_shiny, 11
HTML, 8
htmlOutput, 6, 18
htPositionsOnDevice, 7, 26, 27
htShiny, 8, 12
htShinyExample, 10, 11, 11
interactivate, 12
interactivate.DESeqDataSet, 13
interactivate.kde, 13
interactivateDensity2D, 14
InteractiveComplexHeatmapModal, 15
InteractiveComplexHeatmapOutput, 8, 9, 15, 16, 17, 19, 20, 22
InteractiveComplexHeatmapWidget, 19
is_in_sub_heatmap, 21
kde, 14
makeInteractiveComplexHeatmap, 3–5, 16, 18, 20, 21
observe, 16, 20, 25
observeEvent, 16, 20, 25
originalHeatmapOutput, 6, 23, 28
plotOutput, 3–5
rand_mat, 24
record_observation, 25
renderPlot, 22
results, 13
selectArea, 26
selectPosition, 27
shinyApp, 9
subHeatmapOutput, 6, 23, 28
unit, 4–6, 26, 27