Package ‘debrowser’

January 14, 2017

Type Package

Title debrowser: Interactive Differential Expression Analysis Browser

Version 1.2.4

Date 2016-07-05

Author Alper Kucukural <alper.kucukural@umassmed.edu>, Nicholas Merowsky <nicholas.merowsky@umassmed.edu>, Manuel Garber <manuel.garber@umassmed.edu>

Maintainer Alper Kucukural <alper.kucukural@umassmed.edu>

Description Bioinformatics platform containing interactive plots and tables for differential gene and region expression studies. Allows visualizing expression data much more deeply in an interactive and faster way. By changing the parameters, user can easily discover different parts of the data that like never have been done before. Manually creating and looking these plots takes time. With this system users can prepare plots without writing any code. Differential expression, PCA and clustering analysis are made on site and the results are shown in various plots such as scatter, bar, box, volcano, ma plots and Heatmaps.

Depends R (>= 3.3.0), shiny, ggvis, jsonlite, shinyjs

License GPL-3 + file LICENSE

LazyData true

Imports DT, ggplot2, RColorBrewer, annotate, ggplot, AnnotationDbi, DESeq2, DOSE, igraph, grDevices, graphics, stats, utils, GenomicRanges, IRanges, S4Vectors, SummarizedExperiment, stringi, reshape2, baySeq, d3heatmap, org.Hs.eg.db, org.Mm.eg.db, limma, edgeR, clusterProfiler, V8, methods, sva

RoxygenNote 5.0.1

Suggests testthat, rmarkdown, knitr, R.rsp

VignetteBuilder knitr, R.rsp

URL https://github.com/UMMS-Biocore/debrowser

BugReports https://github.com/UMMS-Biocore/debrowser/issues/new

biocViews Sequencing, ChIPSeq, RNASeq, DifferentialExpression, GeneExpression, Clustering

NeedsCompilation no
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionButton</td>
<td>4</td>
</tr>
<tr>
<td>addID</td>
<td>5</td>
</tr>
<tr>
<td>add_title_pos</td>
<td>5</td>
</tr>
<tr>
<td>all2all</td>
<td>6</td>
</tr>
<tr>
<td>applyFilters</td>
<td>6</td>
</tr>
<tr>
<td>applyFiltersToMergedComparison</td>
<td>7</td>
</tr>
<tr>
<td>clusterData</td>
<td>8</td>
</tr>
<tr>
<td>compareClust</td>
<td>8</td>
</tr>
<tr>
<td>correctBatchEffect</td>
<td>9</td>
</tr>
<tr>
<td>deServer</td>
<td>9</td>
</tr>
<tr>
<td>deUI</td>
<td>10</td>
</tr>
<tr>
<td>drawPCAExplained</td>
<td>11</td>
</tr>
<tr>
<td>getAfterLoadMsg</td>
<td>11</td>
</tr>
<tr>
<td>getColors</td>
<td>12</td>
</tr>
<tr>
<td>getColorShapeSelection</td>
<td>12</td>
</tr>
<tr>
<td>getCompSelection</td>
<td>13</td>
</tr>
<tr>
<td>getConditionSelector</td>
<td>13</td>
</tr>
<tr>
<td>getCondMsg</td>
<td>14</td>
</tr>
<tr>
<td>getCutOffSelection</td>
<td>14</td>
</tr>
<tr>
<td>getDataForTables</td>
<td>15</td>
</tr>
<tr>
<td>getDataPrepPanel</td>
<td>16</td>
</tr>
<tr>
<td>getDensityPlot</td>
<td>16</td>
</tr>
<tr>
<td>getDomains</td>
<td>17</td>
</tr>
<tr>
<td>getDown</td>
<td>17</td>
</tr>
<tr>
<td>getDownloadSection</td>
<td>18</td>
</tr>
<tr>
<td>getEnrichDO</td>
<td>18</td>
</tr>
<tr>
<td>getEnrichGO</td>
<td>19</td>
</tr>
<tr>
<td>getEnrichKEGG</td>
<td>20</td>
</tr>
<tr>
<td>getGeneList</td>
<td>20</td>
</tr>
<tr>
<td>getGeneSetData</td>
<td>21</td>
</tr>
<tr>
<td>getGOLeftMenu</td>
<td>21</td>
</tr>
<tr>
<td>getGOPlots</td>
<td>22</td>
</tr>
<tr>
<td>getHelpButton</td>
<td>23</td>
</tr>
<tr>
<td>getHoverPlots</td>
<td>23</td>
</tr>
<tr>
<td>getInitialMenu</td>
<td>24</td>
</tr>
<tr>
<td>getIntHeatmap</td>
<td>25</td>
</tr>
<tr>
<td>getIntHeatmapVis</td>
<td>25</td>
</tr>
<tr>
<td>getIQRPlot</td>
<td>26</td>
</tr>
<tr>
<td>getLeftMenu</td>
<td>26</td>
</tr>
<tr>
<td>getLegendSelect</td>
<td>26</td>
</tr>
<tr>
<td>getLoadingMsg</td>
<td>27</td>
</tr>
<tr>
<td>getLogo</td>
<td>27</td>
</tr>
<tr>
<td>getMainPanel</td>
<td>28</td>
</tr>
<tr>
<td>getMainPanelPlots</td>
<td>28</td>
</tr>
<tr>
<td>getMean</td>
<td>29</td>
</tr>
<tr>
<td>getMergedComparison</td>
<td>30</td>
</tr>
<tr>
<td>getMethodDetails</td>
<td>30</td>
</tr>
<tr>
<td>getMostVariedList</td>
<td>31</td>
</tr>
<tr>
<td>getNormalizedMatrix</td>
<td>31</td>
</tr>
<tr>
<td>R topics documented:</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>getOrganism</td>
<td>32</td>
</tr>
<tr>
<td>getOrganismBox</td>
<td>33</td>
</tr>
<tr>
<td>getOrganismPathway</td>
<td>33</td>
</tr>
<tr>
<td>getPCExplained</td>
<td>34</td>
</tr>
<tr>
<td>getPCselection</td>
<td>34</td>
</tr>
<tr>
<td>getProgramTitle</td>
<td>35</td>
</tr>
<tr>
<td>getQCLeftMenu</td>
<td>35</td>
</tr>
<tr>
<td>getQCPanel</td>
<td>36</td>
</tr>
<tr>
<td>getQCPlots</td>
<td>36</td>
</tr>
<tr>
<td>getQCReplot</td>
<td>37</td>
</tr>
<tr>
<td>getSampleNames</td>
<td>38</td>
</tr>
<tr>
<td>getSamples</td>
<td>38</td>
</tr>
<tr>
<td>getSearchData</td>
<td>39</td>
</tr>
<tr>
<td>getSelectedCols</td>
<td>39</td>
</tr>
<tr>
<td>getSelectedDatasetInput</td>
<td>40</td>
</tr>
<tr>
<td>getSelectInputBox</td>
<td>40</td>
</tr>
<tr>
<td>getSelHeat</td>
<td>41</td>
</tr>
<tr>
<td>getShapeColor</td>
<td>42</td>
</tr>
<tr>
<td>getStartPlotsMsg</td>
<td>42</td>
</tr>
<tr>
<td>getStartupMsg</td>
<td>43</td>
</tr>
<tr>
<td>getTableStyle</td>
<td>43</td>
</tr>
<tr>
<td>getTextOnOff</td>
<td>44</td>
</tr>
<tr>
<td>getToolTipPCA</td>
<td>44</td>
</tr>
<tr>
<td>getToolTipText</td>
<td>45</td>
</tr>
<tr>
<td>getUp</td>
<td>45</td>
</tr>
<tr>
<td>getUpDown</td>
<td>46</td>
</tr>
<tr>
<td>hideObj</td>
<td>46</td>
</tr>
<tr>
<td>installpack</td>
<td>47</td>
</tr>
<tr>
<td>link_brush</td>
<td>47</td>
</tr>
<tr>
<td>load_data</td>
<td>48</td>
</tr>
<tr>
<td>logSliderJScode</td>
<td>48</td>
</tr>
<tr>
<td>mainScatter</td>
<td>49</td>
</tr>
<tr>
<td>MAPlot</td>
<td>49</td>
</tr>
<tr>
<td>MAZoom</td>
<td>50</td>
</tr>
<tr>
<td>panel.cor</td>
<td>50</td>
</tr>
<tr>
<td>panel_hist</td>
<td>51</td>
</tr>
<tr>
<td>plot_pca</td>
<td>52</td>
</tr>
<tr>
<td>prepAddQCPlots</td>
<td>53</td>
</tr>
<tr>
<td>prepDataContainer</td>
<td>53</td>
</tr>
<tr>
<td>prepDataForQC</td>
<td>54</td>
</tr>
<tr>
<td>prepDEOutput</td>
<td>54</td>
</tr>
<tr>
<td>push</td>
<td>55</td>
</tr>
<tr>
<td>readMetaData</td>
<td>55</td>
</tr>
<tr>
<td>removeCols</td>
<td>56</td>
</tr>
<tr>
<td>round_vals</td>
<td>56</td>
</tr>
<tr>
<td>runBayseq</td>
<td>57</td>
</tr>
<tr>
<td>runDE</td>
<td>57</td>
</tr>
<tr>
<td>runDESeq</td>
<td>58</td>
</tr>
<tr>
<td>runDESeq2</td>
<td>59</td>
</tr>
<tr>
<td>runEdgeR</td>
<td>60</td>
</tr>
<tr>
<td>runHeatmap</td>
<td>61</td>
</tr>
<tr>
<td>runLimma</td>
<td>61</td>
</tr>
</tbody>
</table>
**Description**

Creates an action button whose value is initially zero, and increments by one each time it is pressed.

**Usage**

```r
actionButton(inputId, label, styleclass = "", size = "", block = FALSE,
icon = NULL, css.class = ",", ...)```

**Arguments**

- `inputId` Specifies the input slot that will be used to access the value.
- `label` The contents of the button—usually a text label, but you could also use any other HTML, like an image.
- `styleclass` The Bootstrap styling class of the button—options are primary, info, success, warning, danger, inverse, link or blank
- `size` The size of the button—options are large, small, mini
- `block` Whether the button should fill the block
- `icon` Display an icon for the button
- `css.class` Any additional CSS class one wishes to add to the action button
- `...` Other argument to feed into shiny::actionButton

**Examples**

```r
actionButton("goDE", "Go to DE Analysis!")```
**addID**

**Description**
Adds an id to the data frame being used.

**Usage**
```
addID(data = NULL)
```

**Arguments**
- `data`, loaded dataset

**Value**
data

**Examples**
```
x <- addID()
```

---

**add_title_pos**

**Description**
Adds a title with extra axis to ggvis plot and sets the positions

**Usage**
```
add_title_pos(vis, ..., title = "Plot Title", align = "left", angle = 0,
dx = 0, dy = 0)
```

**Arguments**
- `vis`, a ggvis plot
- `...`, any additional arguments
- `title` for the plot
- `align` position of the title c("left","right")
- `angle` of the labels in x axis
- `dx`, relative x position of the labels in the x axis
- `dy`, relative y position of the labels in the x axis

**Value**
deseq2 results
Examples

```r
require(ggvis)
mtcars %>%
ggvis(x=\text{cyl}, y=\text{wt}, fill=\text{mpg}) %>%
group_by(\text{mpg}) %>%
layer_bars() %>%
add_title_pos(title = "title", angle=310, dy=0, dx=0) %>%
set_options(width = 400, height = 350)
```

---

**Description**

Prepares all2all scatter plots for given datasets.

**Usage**

```r
all2all(data, cex = 2)
```

**Arguments**

- `data`, data that have the sample names in the header.
- `cex`, text size

**Value**

all2all scatter plots

**Examples**

```r
plot<-all2all(mtcars)
```

---

**Description**

Applies filters based on user selected parameters to be displayed within the DEBrowser.

**Usage**

```r
applyFilters(filt_data = NULL, cols = NULL, conds = NULL, input = NULL)
```
applyFiltersToMergedComparison

Arguments
filt_data, loaded dataset
cols, selected samples
conds, selected conditions
input, input parameters

Value
data

Examples
x <- applyFilters()

Description
Gathers the merged comparison data to be used within the DEBrowser.

Usage
applyFiltersToMergedComparison(merged = NULL, nc = NULL, input = NULL)

Arguments
merged, merged data
nc, the number of comparisons
input, input params

Value
data

Examples
x <- applyFiltersToMergedComparison()
**Description**

Gathers the Cluster analysis data to be used within the GO Term plots.

**Usage**

```r
clusterData(dat = NULL)
```

**Arguments**

- `dat`, the data to cluster

**Value**

clustered data

**Note**

- `clusterData`

**Examples**

```r
mycluster <- clusterData()
```

---

**Description**

Compares the clustered data to be displayed within the GO Term plots.

**Usage**

```r
compareClust(dat = NULL, ont = "CC", org = "org.Hs.eg.db", 
  fun = "enrichGO", title = "Ontology Distribution Comparison", 
  pvalueCutoff = 0.01)
```

**Arguments**

- `dat`, data to compare clusters
- `ont`, the ontology to use
- `org`, the organism used
- `fun`, fun
- `title`, title of the comparison
- `pvalueCutoff`, pvalueCutoff
correctBatchEffect

Value

correctBatchEffect

Note

compareClust

Examples

x <- compareClust()

deserver

deServer

Description

Sets up shinyServer to be able to run DEBrowser interactively.

Usage

deServer(input, output, session)
Arguments

input,       input params from UI
output,     output params to UI
session,   session variable

Value

the panel for main plots;

Note

deServer

Examples

deServer

---

deUI  deUI
deUI  deUI

Description

Creates a shinyUI to be able to run DEBrowser interactively.

Usage

deUI()

Value

the panel for main plots;

Note

deUI

Examples

x<-deUI()
**drawPCAEExplained**

**Description**

Creates a more detailed plot using the PCA results from the selected dataset.

**Usage**

```r
drawPCAEExplained(explainedData = NULL)
```

**Arguments**

- `explainedData`, selected data

**Value**

explained plot

**Examples**

```r
x <- drawPCAEExplained()
```

---

**getAfterLoadMsg**

**Description**

Generates and displays the message to be shown after loading data within the DEBrowser.

**Usage**

```r
getAfterLoadMsg()
```

**Value**

return After Load Msg

**Note**

getAfterLoadMsg

**Examples**

```r
x <- getAfterLoadMsg()
```
getColors

Description
get colors for the domains

Usage
getColors(domains = NULL)

Arguments
domains, domains to be colored

Value
colors

Examples
x <- getColors()

getColorShapeSelection

Description
Generates the fill and shape selection boxes for PCA plots. Metadata file has to be loaded in this case

Usage
g getColorShapeSelection(input = NULL)

Arguments
input, input values

Value
Color and shape selection boxes

Examples
x <- getColorShapeSelection()
getCompSelection

Description
Gathers the user selected comparison set to be used within the DEBrowser.

Usage
getCompSelection(count = NULL)

Arguments
count, comparison count

Note
getCompSelection

Examples
x <- getCompSelection(count = 2)

getConditionSelector

Description
Selects user input conditions to run in DESeq.

Usage
getConditionSelector(num = 0, choices = NULL, selected = NULL)

Arguments
num, panel that is going to be shown
choices, sample list
selected, selected sample list

Examples
x <- getConditionSelector()
getCondMsg

Description
Generates and displays the current conditions and their samples within the DEBrowser.

Usage
getCondMsg(dc = NULL, num = NULL, cols = NULL, conds = NULL)

Arguments
dc, columns
num, selected comparison
cols, columns
conds, selected conditions

Value
return conditions

Note
getCondMsg

Examples
x <- getCondMsg()

getCutOffSelection

Description
Gathers the cut off selection for DE analysis

Usage
gCutOffSelection(nc = 1)

Arguments
nc, total number of comparisons

Value
returns the left menu according to the selected tab;
Note

getCutOffSelection

Examples

x <- getCutOffSelection()

dataForTables

Description

dataForTables get data to fill up tables tab

Usage

dataForTables(input = NULL, init_data = NULL, filt_data = NULL,
  selected = NULL, getMostVaried = NULL, mergedComp = NULL,
  explainedData = NULL)

Arguments

  input,           input parameters
  init_data,      initial dataset
  filt_data,      filt_data
  selected,       selected genes
  getMostVaried,  most varied genes
  mergedComp,     merged comparison set
  explainedData,  pca gene set

Value

data

Examples

x <- getDataForTables()
**getDescription**

Create and show the Condition selection screen to the user within the DEBrowser.

**Usage**

ggetDataPrepPanel(flag = FALSE)

**Arguments**

flag, flag to show the element in the ui

**Value**

returns the left menu according to the selected tab;

**Note**

ggetDataPrepPanel

**Examples**

```r
x <- getDataPrepPanel()
```

**getDescription**

Makes Density plots

**Usage**

ggetDensityPlot(data = NULL, cols = NULL, title = "")

**Arguments**

data, count or normalized data
cols, columns
title, title

**Examples**

```r
ggetDensityPlot()
```
getDomains

Description
Get domains for the main plots.

Usage
getDomains(filt_data = NULL)

Arguments
- filt_data, data to get the domains

Value
domains

Examples
x <- getDomains()

getDown

down regulated data

Description
getDown get down regulated data

Usage
getDown(filt_data = NULL)

Arguments
- filt_data, filt_data

Value
data

Examples
x <- getDown()
getDownloadSection

Description

download section button and dataset selection box in the menu for user to download selected data.

Usage

getDownloadSection(flag = FALSE, type = "main")

Arguments

flag, to show the download selection
type, main vs. QC section

Value

the panel for download section in the menu;

Note

getDownloadSection

Examples

x <- getDownloadSection()

gGetEnrichDO

Description

Gathers the Enriched DO Term analysis data to be used within the GO Term plots.

Usage

gGetEnrichDO(genelist = NULL, pvalueCutoff = 0.01)

Arguments

genelist, gene list
epvalueCutoff, the p value cutoff

Value

enriched DO
### Description

Gathers the Enriched GO Term analysis data to be used within the GO Term plots.

### Usage

```r
getEnrichGO(genelist = NULL, pvalueCutoff = 0.01, org = "org.Hs.eg.db", ont = "CC")
```

### Arguments

- `genelist`, gene list
- `pvalueCutoff`, p value cutoff
- `org`, the organism used
- `ont`, the ontology used

### Value

Enriched GO

### Examples

```r
x <- getEnrichDO()
```
getEnrichKEGG

Description
Gathers the Enriched KEGG analysis data to be used within the GO Term plots.

Usage
getEnrichKEGG(genelist = NULL, pvalueCutoff = 0.01, org = "org.Hs.eg.db")

Arguments
- genelist, gene list
- pvalueCutoff, the p value cutoff
- org, the organism used

Value
Enriched KEGG

Note
getEnrichKEGG

Examples
x <- getEnrichKEGG()

getGeneList

Description
Gathers the gene list to use for GOTerm analysis.

Usage
geneList(genes = NULL, org = "org.Hs.eg.db")

Arguments
- genes, gene list
- org, organism for gene symbol entrez ID conversion

Value
ENTREZ ID list
getGeneSetData

**Note**

GOTerm

geneList symbol to ENTREZ ID conversion

**Examples**

```r
x <- getGeneList(c('OCLN', 'ABCC2'))
```

---

**Description**

Gathers the specified gene set list to be used within the DEBrowser.

**Usage**

```r
getGeneSetData(data = NULL, geneset = NULL)
```

**Arguments**

- `data`, loaded dataset
- `geneset`, given gene set

**Value**

- `data`

**Examples**

```r
x <- getGeneSetData()
```

---

**Description**

Generates the GO Left menu to be displayed within the DEBrowser.

**Usage**

```r
getGOLeftMenu()
```

**Value**

- returns the left menu according to the selected tab;
getGOPlots

Description

Go term analysis panel. Generates appropriate GO plot based on user selection.

Usage

getGOPlots(dataset = NULL, input = NULL)

Arguments

dataset, input,
the dataset used
input params

Note

getGOPlots

Examples

x <- getGOPlots()

getGoPanel

Description

Creates go term analysis panel within the shiny display.

Usage

getGoPanel(flag = FALSE)

Arguments

flag, flag to show the element in the ui

Value

the panel for go term analysis;

Note

getGoPanel

Examples

x <- getGoPanel()
getHelpButton

Value
the panel for go plots;

Note
getGOPlots

Examples
x<- getGOPlots()

getHoverPlots

Description
Prepares the plots going to be shown when a gene hovered in the main plots

Usage
getHoverPlots(barData = NULL, genename = NULL)
getInitialMenu

Arguments

- bardata, barplot data
- genename, gene name in the barplots

Examples

getHoverPlots()

Description

Displays the initial menu within DEBrowser.

Usage

getInitialMenu(input = NULL, output = NULL, session = NULL)

Arguments

- input, input from user
- output, output to user
- session, session info

Value

returns the initial menu

Note

getInitialMenu

Examples

x <- getInitialMenu()
**Description**

GetIntHeatmap

**Usage**

```r
getIntHeatmap(data = NULL, input = NULL, inputQCPlot = NULL)
```

**Arguments**

- `data`, `heatData` 
- `input`, `all input params` 
- `inputQCPlot`, `input params for QC`

**Value**

plot

**Examples**

```r
getIntHeatmap()
```

---

**Description**

Gathers the conditional panel for interactive heatmap

**Usage**

```r
getIntHeatmapVis(randstr = NULL)
```

**Arguments**

- `randstr`, `randstr`

**Value**

the panel interactive heatmap

**Note**

getIntHeatmapVis

**Examples**

```r
x <- getIntHeatmapVis()
```
getIQRPlot

Description
Makes IQR boxplot plot

Usage
getIQRPlot(data = NULL, cols = NULL, title = "")

Arguments
data, count or normalized data
cols, columns
title, title

Examples
getIQRPlot()

getLeftMenu

Description
Generates the left menu for plots within the DEBrowser.

Usage
ggetLeftMenu(input = NULL)

Arguments
input, input values

Value
returns the left menu according to the selected tab;

Note
ggetLeftMenu

Examples
x <- getLeftMenu()
getLegendSelect

Description
select legend

Usage
getLegendSelect()

Note
getLegendSelect

Examples
x <- getLegendSelect()

getLoadingMsg

Description
Creates and displays the loading message/gif to be displayed within the DEBrowser.

Usage
getLoadingMsg()

Value
loading msg

Note
getLoadingMsg

Examples
x <- getLoadingMsg()
getLogo

Description
Generates and displays the logo to be shown within DEBrowser.

Usage
getLogo()

Value
return logo

Note
getLogo

Examples
x <- getLogo()

getMainPanel

Description
main panel for volcano, scatter and maplot. Barplot and box plots are in this page as well.

Usage
getMainPanel(randstr = NULL)

Arguments
randstr, random string for the plot containers

Value
the panel for main plots;

Note
getMainPanel

Examples
x <- getMainPanel()
**getMainPanelPlots**

**Description**

Gathers the plots to be used within the main panel.

**Usage**

```r
getMainPanelPlots(filt_data = NULL, cols = NULL, conds = NULL, 
input = NULL, compselect = NULL)
```

**Arguments**

- `filt_data`, filtered data
- `cols`, selected columns
- `conds`, selected conditions
- `input`, input from ui
- `compselect`, selected comparison number

**Value**

`panel`

**Examples**

```r
x <- getMainPanelPlots()
```

---

**getMean**

**Description**

Gathers the mean for selected condition.

**Usage**

```r
getMean(norm_data = NULL, de_res = NULL, inputconds = NULL, 
colnum = NULL)
```

**Arguments**

- `norm_data`, loaded dataset
- `de_res`, de results
- `inputconds`, input parameters
- `colnum`, colnum
**getMergedComparison**

**Description**
Gathers the merged comparison data to be used within the DEBrowser.

**Usage**
```
getMergedComparison(Dataset = NULL, dc = NULL, nc = NULL, input = NULL)
```

**Arguments**
- **Dataset**, whole data
- **dc**, data container
- **nc**, the number of comparisons
- **input**, input params

**Value**
data

**Examples**
```
x <- getMergedComparison()
```

---

**getMethodDetails**

**Description**
get the detail boxes after DE method selected

**Usage**
```
getMethodDetails(num = 0, input = NULL)
```

**Arguments**
- **num**, panel that is going to be shown
- **input**, user input

**Examples**
``` R
x <- getMethodDetails()
```
### getMostVariedList

**Examples**

```r
x <- getMethodDetails()
```

---

**Description**

Calculates the most varied genes to be used for specific plots within the DEBrowser.

**Usage**

```r
getMostVariedList(datavar = NULL, cols = NULL, topn = 500, mincount = 10)
```

**Arguments**

- datavar, loaded dataset
- cols, selected columns
- topn, most varied records
- mincount, total min read count for selected samples

**Value**

data

**Examples**

```r
x <- getMostVariedList()
```

---

### getNormalizedMatrix

**Description**

Normalizes the matrix passed to be used within various methods within DEBrowser. Requires edgeR package

**Usage**

```r
getNormalizedMatrix(M = NULL, method = "TMM")
```

**Arguments**

- M, numeric matrix
- method, normalization method for edgeR. default is TMM
getOrganism

Value
normalized matrix

Note
getGoPanel

Examples
x <- getNormalizedMatrix(mtcars)

getOrganism

Description
getOrganism

Usage
getOrganism(org)

Arguments
org, organism

Value
organism name for keg

Note
getOrganism

Examples
x <- getOrganism()
getOrganismBox

Description
Get the organism Box.

Usage
getOrganismBox()

Value
selectInput

Note
getOrganismBox
getOrganismBox makes the organism box

Examples
x <- getOrganismBox()

getOrganismPathway

Description
getOrganismPathway

Usage
getOrganismPathway(org)

Arguments
org, organism

Value
organism name for pathway

Note
getOrganismPathway

Examples
x <- getOrganismPathway()
getPCexplained

Description
Create a more detailed plot using the PCA results from the selected dataset.

Usage
getPCexplained(datasetInput = NULL, input = NULL)

Arguments
datasetInput, selected data
input, from user

Value
explained plot

Examples
x <- getPCexplained()

getPCselection

Description
Generates the PC selection number to be used within DEBrowser.

Usage
getPCselection(num = 1, xy = "x")

Arguments
num, PC selection number
xy, x or y coordinate

Value
PC selection for PCA analysis

Note
getPCselection

Examples
x <- getPCselection()
**getProgramTitle**

**Description**
Generates the title of the program to be displayed within DEBrowser. If it is called in a program, the program title will be hidden.

**Usage**
```
getProgramTitle(session = NULL)
```

**Arguments**
- `session`, session var

**Value**
program title

**Note**
getProgramTitle

**Examples**
```
title<-getProgramTitle()
```

---

**getQCLeftMenu**

**Description**
Generates the left menu to be used for QC plots within the DEBrowser.

**Usage**
```
getQCLeftMenu(input = NULL)
```

**Arguments**
- `input`, input values

**Value**
QC left menu

**Note**
getQCLeftMenu
getQCPlots

Examples

```r
x <- getQCLeftMenu()
```

---

getQCPPanel

### Description

Gathers the conditional panel for QC plots

### Usage

```r
getQCPPanel(input = NULL)
```

#### Arguments

- `input`: user input

#### Value

the panel for QC plots

### Note

getQCSection

### Examples

```r
x <- getQCPPanel()
```

---

getQCPlots

### Description

Gathers the plot data to be displayed within the quality checks panel.

### Usage

```r
getcPlots(dataset = NULL, input = NULL, metadata = NULL, 
          inputQCPlot = NULL, drawPCAExplained = NULL)
```

#### Arguments

- `dataset`: the dataset to use
- `input`: user input
- `metadata`: coupled samples and conditions
- `inputQCPlot`: input QC params
- `drawPCAExplained`: to draw pca loading plot
**getQCReplot**

**Value**

the panel for QC plots

**Note**

getQCPlots

**Examples**

```r
x <- getQCPlots()
```

---

**getQCReplot**

**Description**

Prepares QCplots for comparisons and others

**Usage**

```r
getQCReplot(cols = NULL, conds = NULL, datasetInput = NULL, input = NULL, inputQCPlot = NULL, drawPCAExplained = NULL)
```

**Arguments**

- `cols`, the dataset to use
- `conds`, the dataset to use
- `datasetInput`, the dataset to use
- `input`, user input
- `inputQCPlot`, input QC params
- `drawPCAExplained`, to draw pca loading plot

**Value**

the panel for QC plots

**Note**

getQCReplot

**Examples**

```r
x <- getQCReplot()
```
**getSampleNames**

**Description**

Prepares initial samples to fill condition boxes. It reads the sample names from the data and splits into two.

**Usage**

```r
getSampleNames(cnames = NULL, part = 1)
```

**Arguments**

- `cnames`, sample names in the header of a dataset
- `part`, c(1,2). 1=first half and 2= second half

**Value**

sample names.

**Examples**

```r
x <- getSampleNames()
```

**getSamples**

**Description**

Gathers the sample names to be used within DEBrowser.

**Usage**

```r
getsamples(cnames = NULL, index = 2)
```

**Arguments**

- `cnames`, names of the samples
- `index`, starting column in a tab separated file

**Value**

- `choices`

**Examples**

```r
x <- getSamples()
```
**getSearchData**

**Description**

search the geneset in the tables and return it

**Usage**

getSearchData(dat = NULL, input = NULL)

**Arguments**

dat, table data
input, input params

**Value**

data

**Examples**

x <- getSearchData()

---

**getSelectedCols**

**Description**

gets selected columns

**Usage**

getSelectedCols(data = NULL, datasetInput = NULL, input = NULL)

**Arguments**

data, all loaded data
datasetInput, selected dataset
input, user input params

**Examples**

getSelectedCols()
**getSelectedDatasetInput**

**Description**

Gathers the user selected dataset output to be displayed.

**Usage**

```r
getSelectedDatasetInput(rdata = NULL, getSelected = NULL,
getMostVaried = NULL, mergedComparison = NULL, explainedData = NULL,
input = NULL)
```

**Arguments**

- `rdata`, filtered dataset
- `getSelected`, selected data
- `getMostVaried`, most varied data
- `mergedComparison`, merged comparison data
- `explainedData`, pca set
- `input`, input parameters

**Value**

data

**Examples**

```r
x <- getSelectedDatasetInput()
```

---

**getSelectInputBox**

**Description**

Selects user input conditions to run in DESeq.

**Usage**

```r
getSelectInputBox(id = NULL, name = NULL, num = 0, choices = NULL,
selected = NULL)
```
**getSelHeat**

**Arguments**

- **id**: input id
- **name**: label of the box
- **num**: panel that is going to be shown
- **choices**: sample list
- **selected**: selected sample list

**Examples**

```r
x <- getSelectInputBox()
```

**Description**

heatmap selection functionality

**Usage**

```r
getSelHeat(data = NULL, input = NULL)
```

**Arguments**

- **data**: selected genes
- **input**: input params

**Value**

plot

**Examples**

```r
x <- getSelHeat()
```
**getShapeColor**

**Description**
Generates the fill and shape selection boxes for PCA plots. Metadata file has to be loaded in this case.

**Usage**
getShapeColor(input = NULL)

**Arguments**
input, input values

**Value**
Color and shape from selection boxes or defaults

**Examples**
```r
x <- getShapeColor()
```

**getStartPlotsMsg**

**Description**
Generates and displays the starting message to be shown once the user has first seen the main plots page within DEBrowser.

**Usage**
getStartPlotsMsg()

**Value**
return start plot msg

**Note**
getStartPlotsMsg

**Examples**
```r
x <- getStartPlotsMsg()
```
getStartupMsg

Description
Generates and displays the starting message within DEBrowser.

Usage
getStartupMsg()

Value
return startup msg

Note
getStartupMsg

Examples
x <- getStartupMsg()

getTableStyle

Description
User defined selection that selects the style of table to display within the DEBrowser.

Usage
getTableStyle(dat = NULL, input = NULL, padj = c("padj"),
foldChange = c("foldChange"), DEsection = TRUE)

Arguments
dat, dataset
input, input params
padj, the name of the padj value column in the dataset
foldChange, the name of the foldChange column in the dataset
DEsection, if it is in DESection or not

Note
getTableStyle

Examples
x <- getTableStyle()
**getDescription**

**Usage**

generateTextOnOff()

**Note**
generateTextOnOff

**Examples**

```r
x <- generateTextOnOff()
```

---

**getToolTipPCA**

**Description**

Prepares tooltip text for PCA plot

**Usage**

generateToolTipPCA(dat = NULL)

**Arguments**

dat, data

**Value**
tooltip text

**Examples**

```r
x <- generateToolTipPCA()
```
getToolTipText

Description
Prepares tooltiptext for the second scatter plot in the plots page

Usage
getToolTipText(dat = NULL)

Arguments
dat,  data need to have following columns; padj, average, cond1 and cond2 values, log10padj, foldChange

Value
tooltip text

Examples
x <- getToolTipText()

getUp

description
getUp get up regulated data

Usage
getUp(filt_data = NULL)

Arguments
filt_data,  filt_data

Value
data

Examples
x <- getUp()
getUpDown

getUpDown get up+down regulated data

Description
getUpDown get up+down regulated data

Usage
getUpDown(filt_data = NULL)

Arguments
filt_data, filt_data

Value
data

Examples
x <- getUpDown()

hideObj

Hides a shiny object.

Usage
hideObj(btns = NULL)

Arguments
btns, hide group of objects with shinyjs

Examples
x <- hideObj()
installpack

Description
install packages if they don’t exist display.

Usage
installpack(package_name = NULL)

Arguments
package_name, package name to be installed

Note
installpack

Examples
x <- installpack()

link_brush

Description
Modified linked brush object. A link brush function modified to be able to create non-reactive linked brush object for ggvis plots

Usage
link_brush()

Value
A list with components:
input A function that takes a visualisation as an argument and adds an input brush to that plot
selected A reactive providing a logical vector that describes which points are under the brush

Note
link_brush is very new and is likely to change substantially

Examples
lb <- link_brush()
load_data

Description
Loads user selected data to be used for DESeq

Usage
load_data(input = NULL, session = NULL)

Arguments
input, input values
session, if data is going to be loaded from json

Value
data

Examples
x <- load_data()

---

logSliderJScode

Description
Generates the log based slider to be used by the user within DEBrowser.

Usage
logSliderJScode(slidername = NULL)

Arguments
slidername, id of the slider

Value
returns the slider values in log10 scale

Note
logSliderJScode

Examples
x <- logSliderJScode()
**mainScatter**

**Description**

Creates the main scatter plot to be displayed within the main panel.

**Usage**

```r
mainScatter(dat = NULL, lb = NULL, data_tooltip = NULL, x = NULL, y = NULL, domains = NULL, colors = NULL)
```

**Arguments**

dat, dataframe that has log2FoldChange and log10padj values
lb, the linked brush
data_tooltip, tooltip specific to this plot
x, the name of the x coordinate
y, the name of the y coordinate
domains, the domains to be colored
colors, colors for each domain

**Value**

volcano plot

**Examples**

```r
x <- mainScatter()
```

---

**MAPlot**

**Description**

Prepares MA plot to be used within the main plot panel.

**Usage**

```r
MAPlot(dat = NULL, lb = NULL, data_tooltip = NULL, domains = NULL, colors = NULL)
```

**Arguments**

dat, dataframe that has log2FoldChange and log10padj values
lb, the linked brush
data_tooltip, tooltip specific to this plot
domains, the domains to be colored
colors, colors for each domain
### Description
Prepares the zoomed in version of the MA plot to be used within the main panel.

#### Usage
```
MAZoom(dat = NULL, data_tooltip = NULL, domains = NULL, colors = NULL)
```

#### Arguments
- `dat`, dataframe that has log2FoldChange and log10padj values
- `data_tooltip`, tooltip specific to this plot
- `domains`, the domains to be colored
- `colors`, colors for each domain

#### Value
zoomed MA plot

#### Examples
```
x <- MAZoom()
```

---

### Description
Prepares the correlations for the all2all plot.

#### Usage
```
panel.cor(x, y, prefix = "rho=", cex.cor = 2, ...)
```

#### Examples
```
x <- MAZoom()
```
panel.hist

Arguments

- **x**, numeric vector x
- **y**, numeric vector y
- **prefix**, prefix for the text
- **cex.cor**, correlation font size
- **...**, additional parameters

Value

all2all correlation plots

Examples

```r
panel.cor(c(1,2,3), c(4,5,6))
```

panel.hist

Description

Prepares the histogram for the all2all plot.

Usage

```r
panel.hist(x, ...)
```

Arguments

- **x**, a vector of values for which the histogram is desired
- **...**, any additional params

Value

all2all histogram plots

Examples

```r
panel.hist(1)
```
Description

Plots the PCA results for the selected dataset.

Usage

```
plot_pca(dat = NULL, pcx = 1, pcy = 2, metadata = NULL, color = NULL, shape = NULL, size = NULL, textonoff = "Off", legendSelect = "fill")
```

Arguments

dat, data

pcx, x axis label

pcy, y axis label

metadata, additional data

color, color for plot

shape, shape for plot

size, size of the plot

textonoff, text on off

legendSelect, select legend

Value

pca list

Examples

```
load(system.file("extdata", "demo", "demodata.Rda", package="debrowser"))
metadata<-cbind(colnames(demodata[,2:7]), colnames(demodata[,2:7]), c(rep("Cond1",3), rep("Cond2",3)))
colnames(metadata)<-c("samples", "color", "shape")
a <- plot_pca(getNormalizedMatrix(
demodata[rowSums(demodata[,2:7])>10,2:7]),
metadata = metadata, color = "samples", size = 5, shape = "shape")
```
**Description**
prepAddQCPlots prepares IQR and density plots

**Usage**
`prepAddQCPlots(data = NULL, input = NULL)`

**Arguments**
- `data`, barplot data
- `input`, user input params

**Examples**
`prepAddQCPlots()`

---

**Description**
prepDataContainer prepares the data container that stores values used within DESeq.

**Usage**
`prepDataContainer(data = NULL, counter = NULL, input = NULL)`

**Arguments**
- `data`, loaded dataset
- `counter`, the number of comparisons
- `input`, input parameters

**Value**
data

**Examples**
`x <- prepDataContainer()`
Description
Prepares selected data for QC plots.

Usage
prepDataForQC(dataset = NULL)

Arguments
dataset, loaded dataset

Value
data

Examples
x <- prepDataForQC()

Description
Prepares the output data from DE analysis to be used within DEBrowser

Usage
prepDEOutput(data = NULL, cols = NULL, conds = NULL, inputconds = NULL, i = NULL)

Arguments
data, loaded dataset
cols, columns
conds, conds
inputconds, inputconds
i, selected comparison number

Value
data

Examples
x <- prepDEOutput()
push

Description
Push an object to the list.

Usage
push(l, ...)

Arguments
1, that are going to push to the list
..., list object

Value
combined list

Examples
mylist <- list()
newlist <- push ( 1, mylist )

readMetaData

Description
read metadata file

Usage
readMetaData(input = NULL)

Arguments
input, input values

Note
readMetaData

Examples
x <- readMetaData()
removeCols

Description
remove unnecessary columns

Usage
removeCols(cols = NULL, dat = NULL)

Arguments
cols, columns that are going to be removed from data frame
dat, data

Value
data

Examples
x <- removeCols()

round_vals

Description
Plot PCA results.

Usage
round_vals(l)

Arguments
l, the value

Value
round value

Examples
x <- round_vals(5.1323223)
runBayseq

Description
Run Bayseq algorithm on the selected conditions. Output is to be used for the interactive display.

Usage
runBayseq(data = NULL, columns = NULL, conds = NULL, rowsum.filter = 10)

Arguments
data,
A matrix that includes all the expression raw counts, rownames has to be the
gene, isoform or region names/IDs

columns,
is a vector that includes the columns that are going to be analyzed. These
columns has to match with the given data.

conds,
experimental conditions. The order has to match with the column order

rowsum.filter,regions/genes/isoforms with total count (across all samples) below this value
will be filtered out

Value
BaySeq results

Examples
x <- runBayseq()

runDE

Description
Run DE algorithms on the selected parameters. Output is to be used for the interactive display.

Usage
runDE(data = NULL, columns = NULL, conds = NULL, pars = NULL)

Arguments
data,
A matrix that includes all the expression raw counts, rownames has to be the
gene, isoform or region names/IDs

columns,
is a vector that includes the columns that are going to be analyzed. These
columns has to match with the given data.

conds,
experimental conditions. The order has to match with the column order

pars,all params for the de methods
Value
deseq2 results

Examples

```r
x <- runDE()
```

---

**Description**

Run DESeq2 algorithm on the selected conditions. Output is to be used for the interactive display.

**Usage**

```r
runDESeq(data, columns, conds, fitType = c("parametric", "local", "mean"),
          non_expressed_cutoff = 10)
```

**Arguments**

- **data**, A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
- **columns**, is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
- **conds**, experimental conditions. The order has to match with the column order
- **fitType**, DESeq2 fitType, it can be 'parametric', 'local', 'mean'.
- **non_expressed_cutoff**, to remove unexpressed regions/genes/isoforms this cutoff is used

**Value**
deseq2 results

**Examples**

```r
x <- runDESeq(data=NULL, columns=c())
```
**Description**

Run DESeq2 algorithm on the selected conditions. Output is to be used for the interactive display.

**Usage**

```r
rndDESeq2(data = NULL, columns = NULL, conds = NULL,
          fitType = c("parametric", "local", "mean"), betaPrior = 0,
          testType = c("Wald", "LRT"), rowsum.filter = 10)
```

**Arguments**

- `data`, A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
- `columns`, is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
- `conds`, experimental conditions. The order has to match with the column order
- `fitType`, either "parametric", "local", or "mean" for the type of fitting of dispersions to the mean intensity. See estimateDispersions for description.
- `betaPrior`, whether or not to put a zero-mean normal prior on the non-intercept coefficients. See nbinomWaldTest for description of the calculation of the beta prior. By default, the beta prior is used only for the Wald test, but can also be specified for the likelihood ratio test.
- `testType`, either "Wald" or "LRT", which will then use either Wald significance tests (defined by nbinomWaldTest), or the likelihood ratio test on the difference in deviance between a full and reduced model formula (defined by nbinomLRT)
- `rowsum.filter`, regions/genes/isoforms with total count (across all samples) below this value will be filtered out

**Value**

deseq2 results

**Examples**

```r
x <- runDESeq2()
```
runEdgeR

Description
Run EdgeR algorithm on the selected conditions. Output is to be used for the interactive display.

Usage
runEdgeR(data = NULL, columns = NULL, conds = NULL, normfact = c("TMM", "RLE", "upperquartile", "none"), dispersion = 0, testType = c("glmLRT", "exactTest"), rowsum.filter = 10)

Arguments
- data, A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
- columns, is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
- conds, experimental conditions. The order has to match with the column order
- normfact, Calculate normalization factors to scale the raw library sizes. Values can be "TMM","RLE","upperquartile","none".
- dispersion, either a numeric vector of dispersions or a character string indicating that dispersions should be taken from the data object. If a numeric vector, then can be either of length one or of length equal to the number of genes. Allowable character values are "common", "trended", "tagwise" or "auto". Default behavior ("auto") is to use most complex dispersions found in data object.
- testType, exactTest or glmLRT. exactTest: Computes p-values for differential abundance for each gene between two digital libraries, conditioning on the total count for each gene. The counts in each group as a proportion of the whole are assumed to follow a binomial distribution. glmLRT: Fit a negative binomial generalized log-linear model to the read counts for each gene. Conduct genewise statistical tests for a given coefficient or coefficient contrast.
- rowsum.filter, regions/genes/isoforms with total count (across all samples) below this value will be filtered out

Value
edgeR results

Examples
x <- runEdgeR()
runHeatmap

Description

Creates a heatmap based on the user selected parameters within shiny.

Usage

runHeatmap(data, title = "Title", dend = "both", names = FALSE,
clustering_method = c("ward.D2", "complete", "single", "average",
"mcquitty", "median", "centroid"), distance_method = c("euclidean", "cor",
"maximum", "manhattan", "canberra", "binary", "minkowski"),
interactive = FALSE)

Arguments

data,        a matrix that includes expression values
title,       title of the heatmap
dend,        dendrogram
names,       a flag to show the rownames
clustering_method = c('complete', 'ward.D2', 'single', 'average', 'mcquitty', 'median', 'centroid')
distance_method = c('cor', 'euclidean', 'maximum', 'manhattan', 'canberra', 'binary', 'minkowski')
interactive, interactive heatmap

Value

heatmap.2 plot

Examples

x <- runHeatmap(mtcars)

runLimma

Description

Run Limma algorithm on the selected conditions. Output is to be used for the interactive display.

Usage

runLimma(data = NULL, columns = NULL, conds = NULL, normfact = c("none",
"TMM", "RLE", "upperquartile"), fitType = c("ls", "robust"),
normBet = c("none", "scale", "quantile", "cyclicloess", "Aquantile",
"Gquantile", "Rquantile", "Tquantile"), rowsum.filter = 10)
Arguments

- **data**, A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
- **columns**, a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
- **conds**, experimental conditions. The order has to match with the column order
- **normfact**, Calculate normalization factors to scale the raw library sizes. Values can be "TMM","RLE","upperquartile","none".
- **fitType**, fitting method; "ls" for least squares or "robust" for robust regression
- **normBet**, Normalizes expression intensities so that the intensities or log-ratios have similar distributions across a set of arrays.
- **rowsum.filter**, regions/genes/isoforms with total count (across all samples) below this value will be filtered out

Value

Limma results

Examples

```r
x <- runLimma()
```

---

run_pca

Description

Runs PCA on the selected dataset.

Usage

```r
run_pca(x = NULL, retx = TRUE, center = TRUE, scale = TRUE)
```

Arguments

- **x**, dataframe with experiment data
- **retx**, specifies if the data should be returned
- **center**, center the PCA (Boolean)
- **scale**, scale the PCA (Boolean)

Value

pca list
**saveQCPlot**

**Examples**

```r
load(system.file("extdata", "demo", "demodata.Rda", package="debrowser"))
pca_data<-run_pca(getNormalizedMatrix(demodata[rowSums(demodata[,2:7])>10,2:7]))
```

**Description**

Saves the current QC plot selection to the users local disk.

**Usage**

```r
saveQCPlot(filename = NULL, input = NULL, datasetInput = NULL, cols = NULL, conds = NULL, inputQCPlot = NULL)
```

**Arguments**

- `filename`: filename
- `input`: input params
- `datasetInput`: dataset
- `cols`: selected columns
- `conds`: selected conditions
- `inputQCPlot`: clustering method and distance method

**Note**

`saveQCPlot`

**Examples**

```r
saveQCPlot()
```
Description

Displays the zoomed in version of the plot to be viewed within the main panel.

Usage

scatterZoom(dat = NULL, data_tooltip = NULL, x = NULL, y = NULL,
domains = NULL, colors = NULL)

Arguments

dat, dataframe that has log2FoldChange and log10padj values
data_tooltip, tooltip specific to this plot
x, the name of the x coordinate
y, the name of the y coordinate
domains, the domains to be colored
colors, colors for each domain

Value

zoomed scatter plot

Examples

x <- scatterZoom()

selectBatchEffect

Description

Batch effect column selection

Usage

selectBatchEffect(input = NULL, selectname = "batchselect",
label = "Batch effect correction column")

Arguments

input, input values
selectname, name of the select box
label, label of the select box
**selectConditions**

**Note**

selectBatchEffect

**Examples**

\[ x <- \text{selectBatchEffect()} \]

---

**selectConditions**

**Description**

Selects user input conditions, multiple if present, to be used in DESeq.

**Usage**

\[ \text{selectConditions(Dataset = NULL, choicecounter, input = NULL)} \]

**Arguments**

- Dataset, used dataset
- choicecounter, total number of comparisons
- input, input params

**Value**

the panel for go plots;

**Note**

selectConditions

**Examples**

\[ x<- \text{selectConditions()} \]
selectedInput

Description

Selects user input conditions to run in DESeq.

Usage

```
selectedInput(id = NULL, num = 0, default = NULL, input = NULL)
```

Arguments

- **id**, input id
- **num**, panel that is going to be shown
- **default**, default text
- **input**, input params

Examples

```
x <- selectedInput()
```

setFilterParams

Description

It sets the filter parameters

Usage

```
setFilterParams(session = NULL, input = NULL)
```

Arguments

- **session**, session variable
- **input**, input parameters

Examples

```
x <- setFilterParams()
```
showObj

Description
Displays a shiny object.

Usage
`showObj(btns = NULL)`

Arguments
- `btns`, show group of objects with shinyjs

Examples
`x <- showObj()`

startDEBrowser

Description
Starts the DEBrowser to be able to run interactively.

Usage
`startDEBrowser()`

Value
the app

Note
`startDEBrowser`

Examples
`startDEBrowser()`
textareaInput

Description
Generates a text area input to be used for gene selection within the DEBrowser.

Usage
textareaInput(id, label, value, rows = 20, cols = 35, class = "form-control")

Arguments
- id, id of the control
- label, label of the control
- value, initial value
- rows, the # of rows
- cols, the # of cols
- class, css class

Examples
x <- textareaInput("genesetarea", "Gene Set", "Fgf21", rows = 5, cols = 35)

togglePanels

Description
User defined toggle to display which panels are to be shown within DEBrowser.

Usage
togglePanels(num = NULL, nums = NULL, session = NULL)

Arguments
- num, selected panel
- nums, all panels
- session, session info

Note
togglePanels

Examples
x <- togglePanels()
volcanoPlot

Description
Prepares volcano plot to be used within the DEBrowser.

Usage
volcanoPlot(dat = NULL, lb = NULL, data_tooltip = NULL, domains = NULL, colors = NULL)

Arguments
dat, dataframe that has log2FoldChange and log10padj values
lb, the linked brush
data_tooltip, tooltip specific to this plot
domains, the domains to be colored
colors, colors for each domain

Value
volcano plot

Examples
x <- volcanoPlot()

volcanoZoom

Description
Prepares the zoomed in version of the volcano plot to be used within the Debrowser.

Usage
volcanoZoom(dat = NULL, data_tooltip = NULL, domains = NULL, colors = NULL)

Arguments
dat, dataframe that has log2FoldChange and log10padj values
data_tooltip, tooltip specific to this plot
domains, the domains to be colored
colors, colors for each domain
Value

zoomed volcano plot

Examples

```
x <- volcanoZoom()
```
Index

actionButton, 4
add_title_pos, 5
addID, 5
all2all, 6
applyFilters, 6
applyFiltersToMergedComparison, 7
clusterData, 8
compareClust, 8
correctBatchEffect, 9
deServer, 9
deUI, 10
drawPCAExplained, 11
getAfterLoadMsg, 11
getColors, 12
g getColorShapeSelection, 12
g getCompSelection, 13
g getConditionSelector, 13
g getCondMsg, 14
g getCutOffSelection, 14
g getDataForTables, 15
g getDataPrepPanel, 16
g getDensityPlot, 16
g getDomains, 17
g getDown, 17
g getDownloadSection, 18
g getEnrichDO, 18
g getEnrichGO, 19
g getEnrichKEGG, 20
g getGeneList, 20
g getGeneSetData, 21
g getGOLeftMenu, 21
g getGoPanel, 22
g getGQPlots, 22
g getHelpButton, 23
g getHoverPlots, 23
g getInitialMenu, 24
g getIntHeatmap, 25
g getIntHeatmapVis, 25
g getIQRPlot, 26
g getLeftMenu, 26
g getLegendSelect, 27
g getLoadingMsg, 27
g getLogo, 28
g getMainPanel, 28
g getMainPanelPlots, 29
g getMean, 29
g getMergedComparison, 30
g getMethodDetails, 30
g getMostVariedList, 31
g getNormalizedMatrix, 31
g getOrganism, 32
g getOrganismBox, 33
g getOrganismPathway, 33
g getPCAExplained, 34
g getPCselection, 34
g getProgramTitle, 35
g getQCLeftMenu, 35
g getQCPanel, 36
g getQCPlots, 36
g getQCreplot, 37
g getSampleNames, 38
g getSamples, 38
g getSearchData, 39
g getSelectedCols, 39
g getSelectedDatasetInput, 40
g getSelectInputBox, 40
g getSelHeat, 41
g getShapeColor, 42
g getStartPlotsMsg, 42
g getStartupMsg, 43
g getTableStyle, 43
g getTextOnOff, 44
g getToolTipPCA, 44
g getToolTipText, 44
g getUp, 45
g getUpDown, 46
hideObj, 46
installpack, 47
link_brush, 47
load_data, 48
logSliderJScode, 48
mainScatter, 49
MAPlot, 49
MAZoom, 50

panel.cor, 50
panel.hist, 51
plot_pca, 52
prepAddQCPlots, 53
prepDataContainer, 53
prepDataForQC, 54
prepDEOutput, 54
push, 55

readMetaData, 55
removeCols, 56
round_vals, 56
run_pca, 62
runBayseq, 57
runDE, 57
runDESeq, 58
runDESeq2, 59
runEdgeR, 60
runHeatmap, 61
runLimma, 61

saveQCPPlot, 63
scatterZoom, 64
selectBatchEffect, 64
selectConditions, 65
selectedInput, 66
setFilterParams, 66
showObj, 67
startDEBrowser, 67

textareaInput, 68
togglePanels, 68

volcanoPlot, 69
volcanoZoom, 69